النظرية المتقدمة في الحوسبة والأوتوماتية واللغات الشكلية

النظرية المتقدمة في الحوسبة والأوتوماتية واللغات الشكلية

الدكتور

أبو بكر أحمد السيد

قسم علوم الحاسوب

جامعة الكويت

منشورات

ذات السلاسل

الكويت

أبو بكر أحمد السيد.

النظرية المتقدمة في الحوسبة والأوتوماتية واللغات الشكلية. ط1.

الكويت: ذات السلاسل، 2016

219 ص ؛ 5, 16 سم.

ردمك: 978-99966-80-22-9

رقم الإيداع: 2016/0644

الطبعة الأولى

١٤٣٧ هـ - ٢٠١٦م

الناشر: ذات السلاسل للطباعة والنشر والتوزيع

E-mail: ths@thatalsalasil.com.kw
Web site:www.thatalsalasil.com.kw

الكويت - ص.ب: 12041 الشامية 71651

تلفون: 22466266/55 (965+)

فاكس: 22438304 (965+)

@THATALSALASIL

@THATALSALASIL

thatalsalasilbookstore

بسم الله الرحمن الرحيم

المقدمـــة

الحمد لله رب العالمين ، والصلاة والسلام على خاتم المرسلين نبينا محمد وعلى آله وصحبه أجمعين .

اللسان العربي . . شعار الإسلام وأهله

لقد مَيَّز الله تعالى اللغة العربية بخاصة الإبانة والتعبير ، وأنزل بها كتابه الخاتم بلسان عربي مبين ، ﴿ وإنه لتنزيل رب العالمين . نزل به الروح الأمين . على قلبك لتكون من المنذرين . بلسان عربي مبين﴾(*) فاللسان العربي شعار الإسلام وأهله ، واللغة من أعظم شعائر الأمم التي بها يتميزون . والأمة العزيزة تعتز بلغتها وتحرص على استقلالها اللغوي كما تحرص على استقلالها العسكري والاقتصادي سواء . والأمة الذليلة تفرط في لغتها حتى تصبح أجنبية عنها وهي منسوبة إليها . وما ذلت لغة شعب إلا ذل ، وما انحطت إلا كان أمره في ذهاب وإدبار . ومن هنا يَفرض الأجنبي المستعمر لغته فرضاً على الأمة المستعمَرة ، ويركبهم بها ، ويشعرهم عظمته فيها ، ويحبس لغتهم في لغته سجناً مؤبداً ، ويحكم على ماضيهم بالقتل محواً ونسياناً ، ويقيد مستقبلهم في الأغلال التي يصنعها ، فأمرهم من بعدها لأمره تَبع .

العربية . . وشخصية الأمة الإسلامية

فشخصية الأمة الإسلامية تبدو جليَّةً عند اعتيادها الأخذَ بلغة كتابها العزيز ، واستخدامها في شتى الميادين ، وجعلها لغة العلوم والثقافة والمجالات الهامة في حياتها وأمور دينها ودنياها . فإن أهملت الأمة تلك اللغة ، واستبدلت بها غيرها من لغات أجنبية ، فذلك مضيعة لشخصيتها ، وذوبان لتلك الشخصية في كيان الأمم التي تأخذ لغاتها ، واعلم حينئذ أنها قد أصبحت أمة لا شخصية لها ، وعدَّها في الموتى .

(*) سورة الشعراء : ١٩٢ – ١٩٥

" لا جرم كانت لغة الأمة هي الهدف الأول للمستعمرين ، فلن يتحول الشعب أول ما يتحول إلا من لغته ، إذ يكون منشأ التحول من أفكاره وعواطفه وآماله ، وهو إذا انقطع من نسب لغته انقطع من نسب ماضيه ، فليس كاللغة نَسبٌ للعاطفة والفكر ، حتى إن أبناء الأب الواحد لو اختلفت ألسنتهم فنشأ منهم ناشئ على لغة ، ونشأ الثاني على أخرى ، والثالث على لغة ثالثة ، لكانوا في العاطفة كأبناء ثلاثة آباء " (*) .

وكان علماء الأمة رحمهم الله في صدرها الأول على وعي كامل بأثر اللغة في تكوين الأمة ، وخطرها في بناء شخصية المسلم ، ولذلك حرصوا حرصاً شديدا على لغة القرآن والسنة ، وشددوا النكير على من حاد عنها إلى غيرها ، واستبدل الذي هو أدنى بالذي هو خير . وما زال السلف يكرهون تغيير شعائر العرب حتى في المعاملات ، وهو " التكلم بغير العربية " إلا لحاجة .. مع أن سائر الألسن يجوز النطق بها لأصحابها ، ولكن سوَّغوها للحاجة ، وكرهوها لغير الحاجة ، ولحفظ الإسلام .

الله أنزل كتابه باللسان العربي ، وبعث به نبيه العربي ، وجعل الأمة العربية خير الأمم ، فصار حفظ شعارهم من تمام حفظ الإسلام ، فكيف بمن تقدم على الكلام العربي – مفرده ومنظومه – يغيره ويبدله ، ويخرجه عن قانونه ، ويكلف الانتقال عنه ؟

انتشار العربية في بلاد الفتح الإسلامي

ولم يكن العرب عندما يفتحون بلدا من البلدان يتركون لسانهم ولسان قرآنهم من أجل لسان أحد ، وإنما تغلب العربية على أهل المصر المفتوح الذين يألفونها ، بل ويتسابقون في تعلمها ودراستها والتحدث بها ، ويكره المسلمون أشد الكره أن تتفشى فيهم العجمة والرطانة البعيدتان عن لغة الكتاب وأهله .

واعتياد الخطاب بغير العربية حتى يصير ذلك عادة للرجل مع صاحبه ، فلا ريب أن هذا مكروه فإنه من التشبه بالأعاجم . ولهذا كان المسلمون المتقدمون ، لما سكنوا أرض الشام ومصر ، ولغة أهلهما روميَّة ، وأرض العراق وخراسان ، ولغة أهلهما فارسية ،

(*) وحي القلم لمصطفى صادق الرافعي ٣٢/٣

وأرض المغرب ، ولغة أهلها بربرية : عودوا أهل هذه البلاد اللغة العربية ، حتى غلبت على أهل هذه الأمصار : مسلمهم وكافرهم .

ومعلوم كذلك أن اللغة العربية كانت هي لغة العلم والثقافة في بلاد الأندلس أيام الحضارة الإسلامية العالمية الزاهرة .

سيادة اللغة العربية

وترجع سيادة اللغة العربية للغات العالم كله إلى أن الكتاب الخاتم نزل بها ، وتكفل الله عز وجل بحفظه ، فقال تعالى: ﴿ إنا نحن نزلنا الذكر وإنا له لحافظون ﴾ (*)، فهي لغة محفوظة ، وهذا هو السر الذي يجعلنا لا نقيس العربية الفصحى بما يحدث في اللغات الحية المعاصرة ، فإن أقصى عمر هذه اللغات في شكلها الحاضر لا يتعدى قرنين من الزمان ، فهي دائمة التطور والتغير ، وعرضة للتفاعل مع اللغات المجاورة ، تأخذ منها وتعطي ، ولا تجد في كل ذلك حرجا ، لأنها لم ترتبط في فترة من فترات حياتها بكتاب كريم ، كما هي الحال في العربية .

محاربة لغة القرآن جزء من المؤامرة على الإسلام

واليوم تواجه العربية حربا شرسة هي جزء من المؤامرة على الإسلام وأهله ، فلقد قام الصراع بين الإسلام من جانب ، والنحل الفاسدة والملل الكافرة من جانب آخر منذ اليوم الأول الذي صدع فيه النبي صلى الله عليه وسلم بأمر ربه في تبليغ رسالته ، ثم لم يزل يشتد إلى يوم الناس هذا . وكان – ولا يزال – من أهم ميادين الصراع بين الإسلام وخصومه اللغة العربية نفسها ، لغة القرآن العظيم ، ولغة النبي الكريم صلى الله عليه وسلم ، حتى إنه ظهرت دعوات مختلفة إلى كتابة العربية بالحروف اللاتينية لمحاولة

(*) الحجر : ٩

اللحاق بالغرب ! ، وإلى استخدام اللهجات العامية بدلا من العربية الفصحى بحجة التيسير على الناس ! ، وإلى إقصاء العربية تماما عن ميدان العلوم بدعوى مواكبة التطور العلمي الحديث ومسايرة الأبحاث ! ، والله يشهد إن المنافقين لكاذبون ، فهم يخططون ويعملون ليل نهار للقضاء على لغة القرآن ، فإن اللغة التي لا تستخدم تموت وتندثر ، ويسهل بعد ذلك القضاء على أمة الإسلام وتغريبها بعد تغريب لسانها وفكرها وثقافتها !

التعريب . . نصرة لدين الله

إن العربية من الدين ، وإن التعريب من أعظم الخطوات لنصرة هذا الدين . وإن تعلم العربية لفهم مقاصد الكتاب والسنة قربة من أَجَلِّ القربات إلى الله عز وجل ، وإن معرفة ما يقيم به المسلم فرضه فرض واجب عليه . واللسان تقارنه أمور أخرى من العلوم والأخلاق ، فإن العادات لها تأثير عظيم فيما يحبه الله وفيما يكرهه ، فلهذا جاءت الشريعة بلزوم عادات السابقين في أقوالهم وأعمالهم ، وكراهة الخروج عنها إلى غيرها من غير حاجة . وإذا كانت اللغة هي خزانة الفكر الإنساني فإن خزائن العربية قد ادخرت من نفيس البيان الصحيح عن الفكر الإنساني وعن النفوس الإنسانية ما يعجز سائر اللغات .

شهداء على الناس

إن الأمة التي نزل بلسانها الكتاب الكريم يجب عليها أن تعمل على نشر دينها ونشر لسانها ونشر عاداتها وعلومها وآدابها بين الأمم الأخرى ، وهي تدعوها إلى ما جاء به نبيُّها من الهدى ودين الحق لتجعل من هذه الأمم أمة واحدة ، دينها واحد ، وقبلتها واحدة ، ولغتها واحدة ، ومقومات شخصيتها واحدة ، ولتكون أمة وسطا ، ويكونوا شهداء على الناس ، فمن أراد أن يدخل في هذه العصبة الإسلامية فعليه أن يعتقد دينها ، ويتبع شريعتها ، ويهتدي بهديها ، ويتعلم لغتها ، ويكون في ذلك كله تبعا لا متبوعا .

جهاد التعريب

من هنا ندرك خطورة الحملة التغريبية العارمة التي تتعرض لها أمتنا لمسخها في كل مجال : من العقيدة والخلق والقانون وأنظمة التعليم والإعلام ، إلى العادات والتقاليد واللباس والطعام والشراب . ولا بد من بذل الجهود من أجل " تعريب الأمة " ، أي اعتماد الأمة كلها – مؤسسات وأفرادا – اللغة العربية الفصحى أداة وحيدة للتعبير في كل أمر ذي بال ، ومن ذلك أمر العلم والتعليم ، وتظهر أهمية هذا المستوى من التعريب إذا قصدنا بالأمة هنا : الأمة الإسلامية كلها وليس العرب فقط ، وهذا هو التحدي الضخم الذي يواجه المسلمين اليوم ، والذي نجح أسلافنا الصالحون في إنجازه حين حملوا الإسلام إلى خارج جزيرة العرب ، وحملوا معه اللغة العربية فأسلمت الشعوب وتعربت .

إن التعريب ميدان واسع من ميادين الجهاد الحضاري ، فالأمم كما تتصارع حضاريا في الميادين السياسية والاقتصادية والعسكرية ، تتصارع أيضا في الميدان اللغوي الذي لا يقل أهمية وآثارا عن الميادين الأخرى ، بل إنه لسلاح عظيم الفعالية في إحراز النصر في ميادين الصراع الأخرى . ومن هنا نفهم السر في الاهتمام العظيم الذي أولته كل الدول الاستعمارية ، غربية أو شرقية ، لنشر لغاتها بين شعوب الأرض.

وفي إطار ذلك الصراع أو الحرب اللغوية ، حوربت العربية بشراسة في موطنها الأصلي وفي كل البلاد الإسلامية ، ولا تزال العربية حتى اليوم في معقلها الأصلي تواجه – على كل صعيد ومستوى – تلك الحرب التي تنفذ اليوم – للأسف – بأيدي أبنائها الذين يخربون بيوتهم بأيديهم . ولذا فإن جهاد التعريب ضد هذه الغارة الشاملة قد بات فرض عين على كل مسلم وعربي ، فالغزو قد وصل إلى عقر الدار . ولكل جهاد رجاله وخططه وأسلحته . وكل من يقعد عن هذا الجهاد ويخلد إلى الأرض فو آثم ، وكيف لا وهو يرى شخصية الأمة تغتال ، ومقوماتها تنتهك ، وعوامل تماسكها تحطم ، ثم يتقاعس ؟ .. أما أولئك الذين يقفون مع العدو في خندق واحد : يؤازرون مخططاته ، ويحرسون تنفيذها ، فهم لا يقلون خيانة عمن يعمل لحساب العدو أيام الحرب العسكرية ، وما يستحقه هؤلاء يستحقه أولئك .

إن القتل الحضاري ذلا ومسخا وتذويبا لأشَدُّ وأنكى من القصف بالراجمات ، ودك البيوت على الرؤوس ، ولا تقبل أي تَعِلات مهما تكن لتبرير الرضى بقتل هوية الأمة والذي نرى آثاره شاخصة في كل مكان .

هذا الكتاب . . والغاية منه

وضمن جهود التعريب في ميدان العلوم الحديثة ظهرت والحمد لله في السنوات الأخيرة بعض الكتب باللغة العربية (مؤلفة أو مترجمة) في مجال علم الحاسوب ، وهذه الكتب وإن كانت قليلة جدا بالنسبة لما هو مطلوب في عملية تعريب العلوم الحديثة للنهوض بأمتنا ، إلا أنها جهد مشكور ونواة طيبة لمكتبة علمية عربية ندعو الله عز وجل أن يبارك فيها لتثرى وتروي ظمأ الطلاب والدارسين والباحثين . وهذا الكتاب إضافة جديدة لهذه المكتبة في موضوع نظرية الحوسبة والأوتوماتية واللغات الشكلية ، وقد اتبعنا في منهجه وترتيب فصوله ومحتوياته المرجع الثالث في قائمة المراجع المذكورة بنهاية الكتاب، وهو من أفضل المراجع الحديثة في هذا الموضوع . ونأمل بذلك أن ينفع الله تعالى بإذنه سبحانه بهذا الكتاب أبناء العربية ، وأن يسد الكتاب نقصا في المكتبة العلمية العربية ، ويكون بذلك لبنة جديدة في صرح تعريب العلوم الحديثة ، وجهدا متواضعاً ندعو الله عز وجل أن يبارك فيه فيثقل ميزان حسناتنا يوم نقف بين يدي المولى عز وجل فيسألنا ماذا قدمنا لغدنا ، وماذا عملنا من أجل النهوض بأمتنا ورفع راية التوحيد بين الخلق أجمعين .

والكتاب يُعَدُّ امتدادا لكتابنا السابق " نظرية الحوسبة والأوتوماتية واللغات الشكلية " (مكتبة الفلاح ، الكويت ٢٠١١) ، الذي تناول بأسلوب سهل وبسيط أسس ومبادئ وقواعد اللغات الشكلية والآلات (الأوتوماتا) والحوسبة ، وبعض التطبيقات والمجالات ذات الصلة ، والكتاب الحالي يتناول الموضوعات المتقدمة في نظرية الحوسبة والأوتوماتية كآلات تيورنج ونماذجها المتنوعة ، والتسلسل الهرمي لِلُّغات الشكلية والآلات ذاتية الحركة، وحدود العمليات الحسابية الخوارزمية ، والنماذج المختلفة للحوسبة ، ودرجات التعقيد الحاسوبية .

وفي نهاية كل فصل من فصول الكتاب يوجد عدد من التمرينات والمسائل التي تعين الطالب بإذن الله على فهم واستيعاب المفاهيم والطرق التي وردت في ذلك الفصل. ويمكن للطالب التأكد من صحة هذا الفهم بالرجوع إلى أجوبة التمرينات في نهاية الكتاب .

ومن الممكن تدريس مادة هذا الكتاب في فصل دراسي واحد لطلاب كليات علوم الحاسوب وهندسة الحاسوب .

شكر وتقدير

نود هنا أن نشكر الأخت الفاضلة السيدة / سالمة موسى بمكتب نائب مدير الجامعة للشئون العلمية بجامعة الكويت على ما بذلته من جهد كبير وما تحلت به من صبر وأناة في طباعة هذا الكتاب حتى يخرج بهذه الصورة الطيبة التي تخدم عملية التعريب وتيسر نشر العلوم في آفاق أمتنا لتعود كما كانت في مكانة الصدارة بين الأمم . جعل الله ذلك الجهد في ميزان حسناتها لمساهمتها في تيسير سبل طلب العلم والتماسه .

نسأل الله العلي القدير أن ينفع بهذا الكتاب ، وأن يتقبله عملا خالصا لوجهه الكريم ، وعلما نافعا لأبناء أمتنا الإسلامية .

عن معاذ بن جبل رضي الله عنه : (تعلموا العلم ، فإن تعلمه لله خشية ، وطلبه عبادة ، ومذاكرته تسبيح ، والبحث عنه جهاد ، وتعليمه لمن لا يعلمه صدقة ، وبذله لأهله قربة ، لأنه معالم الحلال والحرام ، ومنار سبل أهل الجنة ، وهو الأنيس في الوحشة ، والصاحب في الغربة ، والمحدث في الخلوة ، والدليل على السراء والضراء ، والسلاح على الأعداء ، والزين عند الأخلاء ، يرفع الله به أقواما ، فيجعلهم في الخير قادة وأئمة تقتص آثارهم ، ويقتدى بفعالهم ، وينتهي إلى رأيهم . ترغب الملائكة في خلاتهم ، وبأجنحتها تمسحهم ، ويستغفر لهم كل رطب ويابس ، وحيتان البحر وهوامه ، وسباع البر وأنعامه ، لأن العلم حياة القلوب من الجهل ، ومصابيح الأبصار في الظلم . يبلغ العبد بالعلم منازل الأخيار ، والدرجات العلى في الدنيا والآخرة . التفكير فيه يعدل الصيام ، ومدارسته تعدل القيام ، به توصل الأرحام ، وبه يعرف الحلال من الحرام ، وهو إمام العمل تابعه ، يلهمه السعداء ويحرمه الأشقياء) [ابن عبدالبر]

وآخر دعوانا أن الحمد لله رب العالمين .

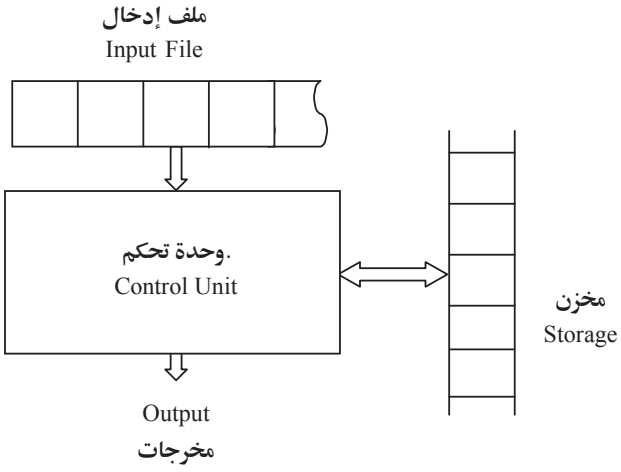
الفصل الأول

آلات تيورنج
Turing Machines

نعلم أن هناك ارتباطا بين اللغات المنتظمة واللغات حرة السياق **(regular and context-free languages)** من ناحية ، والآلات المحدودة (finite automata) وآلات القبول ذوات الدفع لأسفل (pushdown accepters) من ناحية أخرى . ونعلم أن اللغات المنتظمة تكوِّن مجموعة جزئية فعلية / صحيحة (proper subset) من اللغات حرة السياق ، ولذلك فإن آلات الدفع لأسفل تُعد أقوى من الآلات المحدودة . ونعلم كذلك أن اللغات حرة السياق ، رغم أنها أساسية بالنسبة لدراسة لغات البرمجة ، إلا أنها محدودة المجال (limited in scope) . وهذا يتضح لنا عندما يتبين لنا أن بعض اللغات البسيطة مثل $\{a^n b^n c^n\}$ و $\{ww\}$ ليست حرة السياق . وهذا يدفعنا إلى النظر فيما وراء اللغات حرة السياق ، وإلى البحث في كيفية تعريف عائلة لغات جديدة تشتمل على مثل هذه الأمثلة . ومن أجل ذلك فإننا نعود إلى الصورة العامة لأي آلة . إذا قارنا بين الآلات المحدودة والآلات ذوات الدفع لأسفل (pushdown automata) ، فإننا نرى أن طبيعة التخزين المؤقت (temporary storage) هي التي تُنشئ الفارق بينهما . فإذا لم يكن هناك أي تخزين (no storage) فستكون لدينا آلة محدودة . وإن كانت وسيلة التخزين (أي المخزن) رَصَّة (a stack) فستكون لدينا الآلة الأقوى ذات الدفع لأسفل . واستكمالا لهذه الملاحظة يمكننا أن نتوقع أن نكتشف عائلات لغات أقوى إذا أعطينا الآلة وسيلة تخزين أكثر مرونة (more flexible storage). فمثلا ماذا يحدث إذا استخدمنا – في المخطط العام (شكل ١ – ١) رصتين أو ثلاث رصات أو طابورا (a queue) أو أي وسيلة تخزين أخرى ؟ هل كل وسيلة تخزين تُعَرِّف نوعا جديدا من الآلات ومن خلاله عائلة لغات جديدة ؟ ماذا يمكننا أن نقول عن أقوى الآلات وعن حدود الحوسبة ؟ هذا يقودنا إلى المفهوم الأساسي لآلة "تيورنج" (a Turing Machine) ، وبالتالي إلى تعريف دقيق لفكرة حوسبة ميكانيكية أو خوارزمية (a mechanical or algorithmic computation) .

ونبدأ دراستنا بإذن الله بإعطاء تعريف رسمي / شكلي لآلة تيورنج ، ثم نعطي بعض الأمثلة لبرامج بسيطة ، ومن ثَمَّ نزعم أنه رغم أن آلية أو ميكانيكية (mechanism) آلة تيورنج بدائية وبسيطة جدا إلا أن مفهومها واسع جدا بحيث يشمل عمليات معقدة . ونصل بدراستنا إلى نظرية تيورنج

(Turing thesis) التي تقرر أن أي عملية حسابية (computational process) – كتلك التي تقوم بها الحواسيب اليوم – يمكن أن تنفِّذها آلة من آلات تيورنج .



شكل ١ – ١

مخطط تمثيل آلة عامة ذاتية الحركة
schematic representation of a general automaton

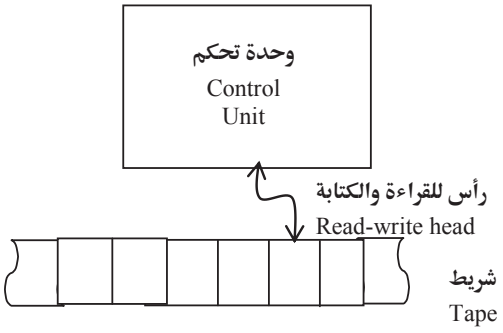
آلة تيورنج القياسية The Standard Turing Machine

تتميز وسيلة التخزين في آلة تيورنج بالبساطة التامة ، حيث يمكن النظر إليها على أنها منظومة مفردة أحادية البعد من خلايا (a single, one-dimensional array of cells) يمكن للواحدة منها أن تحتفظ برمز مفرد . وهذه المنظومة تمتد (extends) بلا حدود (indefinitely) في كلى الاتجاهين ، ولذلك فهي قادرة على الاحتفاظ بكمية غير محدودة من المعلومات . ويمكننا قراءة المعلومات وتغييرها بأي ترتيب . وسنطْلِق على مثل هذه الوسيلة للتخزين شريطا (**a tape**) نظرا لأنها تشبه الشريط المغناطيسي (magnetic tape) الذي كان يستخدم في الحواسيب القديمة .

تعريف آلة تيورنج Definition of a Turing Machine

آلـة تيـورنج هـي آلـة ذاتيـة الحركـة (automaton) ذاكرتهـا / مخزنهـا المؤقـت (temporary storage) عبارة عن شريط . وهذا الشريط مقسَّم إلى خلايا يمكن للواحدة منها الاحتفاظ برمز واحد . وترتبط بالشريط رأس للقراءة والكتابة (read – write head) يمكنها التحرك على الشريط لليمين أو لليسار ، ويمكنها قراءة أو كتابة رمز مفرد (a single symbol) في كل تحـرك . وآلة تيورنـج تختـلف قليلا عن الآلة العامـة ذاتية الحركـة التي سـبق ذكـرها (شكل ١ – ١) في أنها لا تحتوي على ملف إدخال أو أي آلية إخراج خاصة (special output mechanism) . وأي إدخـال أو إخـراج يلزمنا سيتم تنفيـذه على شريط الآلـة (machine's tape) . وسنرى لاحقا بإذن الله أن هذا التعديل على مخطط الآلة العامة له تأثير بسيط . ويمكننا الاحتفاظ بملف الإدخال وآلية إخراج خاصة دون التأثير على أي من الاستنتاجات التي سنصل إليها بعد قليل بإذن الله ، ولكننا سنتركهما لأن الآلة الناتجة – بعد تركهما – تُعَدُّ أسهل قليلا في الوصف .

ويبين شكل ١ –٢ مخططا لآلة تيورنج



شكل ١ – ٢
مخطط لآلة تيورنج

تعريف ١ – ١ : تُعرَّف آلة تيورنج M (Turing machine) على أنها

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$$

حيث

Q : مجموعة الحالات الداخلية (internal states) .

Σ : المجموعة الأبجدية للإدخال (input alphabet) .

Γ : مجموعة محدودة من الرموز (a finite set of symbols) يطلَق عليها المجموعة الأبجدية للشريط (tape alphabet) .

δ : دالة الانتقال (transition function) .

$\square \in \Gamma$: رمز خاص (a special symbol) يطلق عليه الفراغ (blank) .

$q_0 \in Q$: الحالة الابتدائية (initial state) .

$F \subseteq Q$: مجموعة الحالات النهائية (final states) .

وفي هذا التعريف لآلة تيورنج نفترض أن $\Sigma \subset \Gamma - \{\square\}$. أي أن المجموعة الأبجدية للإدخال Σ مجموعة جزئية من المجموعة الأبجدية للشريط عدا الفراغ . وتُستبعد الفراغات كمدخلات لأسباب ستتضح بإذن الله بعد قليل . وتُعرَّف دالة الانتقال δ كما يلي :

$$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$$

وعموما فإن δ دالة جزئية ***على*** $Q \times \Gamma$ (a partial function on) ، وتفسيرها (its interpretation) يعطِي المبدأ الذي تعمل به آلة تيورنج . ووسيطا (arguments) الدالة δ هما الحالة الحالية (current state) لوحدة التحكم ، والرمز الحالي للشريط (current tape symbol) الذي تتم قراءته (being read) . والنتيجة هي حالة جديدة لوحدة التحكم ، ورمز جديد للشريط (والذي يحل محل الرمز القديم) ، ورمز حركة (a move symbol : R أو L . ورمز الحركة هذا يشير إلى اتجاه حركة رأس القراءة والكتابة (لليمين أو لليسار على الترتيب) مسافة خلية واحدة (one cell) بعد إتمام كتابة الرمز الجديد على الشريط .

مثال ١ – ١ :

الشكل التالي (شكل ١ – ٣) يبين الوضع (situation) قبل وبعد تنفيذ الانتقال / الحركة (move) :

$$\delta\,(q_0, a) = (q_1, d, R)$$

(ب)
بعد الحركة

(أ)
قبل الحركة

شكل ١ – ٣

* * *

ويمكننا النظر إلى آلة تيورنج على أنها حاسوب بسيط يحتوي على وحدة تشغيل (a processing unit) لها ذاكرة محدودة (a finite memory) ، وفي شريطه (tape) يحتوي الحاسوب على وحدة تخزين ثانوية (a secondary storage) ذات سعة غير محدودة (unlimited capacity) . والتعليمات التي يستطيع مثل هذا الحاسوب تنفيذها محدودة جدا : فيمكنه الإحساس (sensing) برمز على شريطِهِ ، واستخدام النتيجة لتحديد ماذا يفعل في الخطوة التالية . والخطوات الفعلية التي تستطيع الآلة اتخاذها أو إنجازها هي : إعادة كتابة (rewriting) الرمز الحالي ، وتغيير حالة التحكم (state of control) ، وتحريك رأس القراءة والكتابة . وقد تبدو هذه المجموعة الصغيرة من التعليمات غير مناسبة لإجراء عمليات معقدة ، ولكن الحقيقة أن الأمر ليس كذلك . فآلات تيورنج تتمتع بقدرات فائقة من حيث المبدأ ، ودالة الانتقال δ تُعَرِّف وتحدد كيف يتصرف الحاسوب ، وعادة تُسَمَّى " برنامج الآلة " (program of the machine) .

وكما هو الحال دائما فإن الآلة تبدأ في الحالة الابتدائية المعطاة حيث توجد بعض المعلومات على الشريط . ثم تنتقل الآلة عبر متتابعة من الخطوات التي تحكمها دالة الانتقال δ . وأثناء هذه العملية فإن محتويات أي خلية على الشريط يمكن أن تُفحص وتُغَيَّر عدة مرات . وفي النهاية فإن العملية (process) كلها قد تتوقف / تنتهي (terminate) ، وهي الحالة التي نصل إليها في آلة تيورنج بوضعها في "حالة تَوَقُّف" (**a halt state**) . ويقال إن آلة تيورنج قد توقفت كلما وصلت إلى هيئة / تشكيلة / حالة تكوينية (a configuration) تكون فيها الدالة δ غير معرَّفة ، وهذا شئ ممكن الحدوث نظرا لأن δ دالة جزئية (a partial function) . وفي الحقيقة فإننا سنفترض أنه لا توجد أي انتقالات مُعَرَّفة لأي حالة نهائية (final state) ، وبالتالي فإن آلة تيورنج ستتوقف (halt) كلما دخلت حالة نهائية .

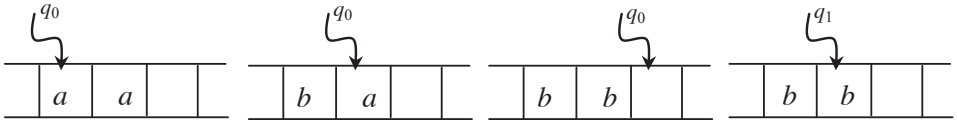
مثال ١ – ٢ :

نفرض أن لدينا آلة تيورنج معرَّفة بما يلي :

$$
\begin{aligned}
Q &= \{q_0, q_1\},\\
\Sigma &= \{a, b\},\\
\Gamma &= \{a, b, \Box\},\\
F &= \{q_1\},\\
\delta\,(q_0, a) &= (q_0, b, R),\\
\delta\,(q_0, b) &= (q_0, b, R),\\
\delta\,(q_0, \Box) &= (q_1, \Box, L).
\end{aligned}
$$

إذا بدأنا آلة تيورنج هذه في الحالة q_0 حيث الرمز a هو الرمز الموجود تحت رأس القراءة والكتابة ، فإن قاعدة الانتقال القابلة للتطبيق هي : $\delta(q_0, a) = (q_0, b, R)$. ولذلك فإن رأس القراءة والكتابة سوف تستبدل بالرمز a رمزاً b ، ثم تتحرك إلى اليمين على الشريط . وستبقى الآلة في الحالة q_0 . وأي رمز a تالي (subsequent) سيُستبدل به أيضا رمز b ، ولكن أي رموز b سوف لا تُعَدَّل . وحينما تصادف الآلة أول فراغ (blank) ، فإنها ستتحرك لليسار خلية واحدة ، ثم تتوقف (halt) في الحالة النهائية q_1 .

ويبين الشكل التالي (شكل ١ – ٤) مراحل (stages) عدة للعملية وذلك بالنسبة لهيئة تكوينية ابتدائية (initial configuration) بسيطة .



شكل ١–٤

متتابعة تحركات

A sequence of moves

* * *

ويمكننا استخدام مخططات بيانية للانتقالات (transition graphs) لتمثيل آلات تيورنج ، حيث نُسمِّي (label) أحرف (edges) المخطط البياني (graph) بثلاثة أسماء /

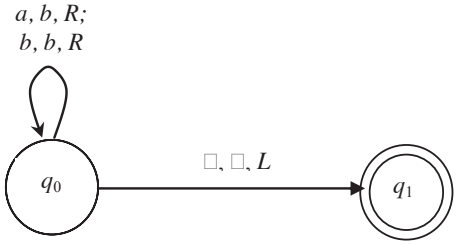
عناوين / عناصر (items) : رمز الشريط الحالي (current tape symbol) ، والرمز الذي سيحل محله (replaces it) ، والاتجاه الذي ستتحرك نحوه رأس القراءة والكتابة .

مثال ١ – ٣ :

ارسم مخطط الانتقالات البياني الذي يمثّل آلة تيورنج المعطاة في مثال ١ – ٢ .

الحل :

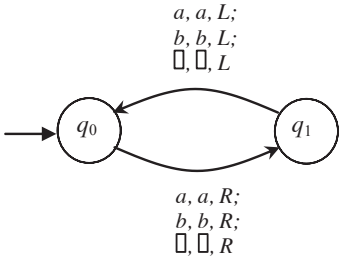
الشكل التالي (شكل ١ – ٥) يعطي مخطط الانتقالات البياني المطلوب .



شكل ١–٥

مثال ١ – ٤ :

نفرض أن لدينا آلة تيورنج يمثلها مخطط الانتقالات البياني التالي (شكل ١ – ٦) :



شكل ١ – ٦

كي نرى ماذا سيحدث مع هذه الآلة يمكننا تتبع حالة نمطية (a typical case) .

نفرض أن الشريط يحتوي ابتداءً على : ... $a\ b$ ، وأن رأس القراءة والكتابة واقفة على الرمز a .

ولذلك فإن الآلة تقرأ الرمز a ، ولكنها لا تغيِّره ، وتنتقل إلى الحالة التالية q_1 ، وتتحرك رأس القراءة

والكتابة إلى اليمين ، بحيث تصبح الآن فوق الرمز b . وهذا الرمز يُقرأ أيضا ويُترك دون تغيير . وتعود الآلة إلى الحالة q_0 ، وتتحرك رأس القراءة والكتابة إلى اليسار . والآن نكون قد عدنا بالضبط إلى الحالة الأصلية، وتبدأ متتابعة التحركات مرة أخرى . ويتضح من هذا أن الآلة – مهما كانت المعلومات الابتدائية على شريطها – ستظل تعمل للأبد ، بحيث تتنقل رأس القراءة والكتابة بين الحركة لليمين والحركة لليسار دون عمل أي تعديلات (modifications) على الشريط . وهذا المثال يُعد مثالا لآلة تيورنج لا تتوقف . وبالمشابهة مع مصطلحات البرمجة نقول إن آلة تيورنج تقع في عروة لا نهائية .

* * *

ونظرا لأنه يمكننا إعطاء عدة تعريفات متنوعة لآلة تيورنج ، فمن المفيد أن نلخص هنا الخصائص الأساسية لنموذجنا ، والذي سنطلق عليه

تعريف ١ – ٢ : " آلة تيورنج القياسية " **" a standard Turing machine "**

(١) تحتوي آلة تيورنج على شريط غير محدود (unbounded) في الاتجاهين ، مما يسمح بأي عدد من التحركات لليسار ولليمين .

(٢) آلة تيورنج آلة محددة (deterministic) بمفهوم أن دالة الانتقال δ تُعَرِّف على الأكثر حركة واحدة لكل تشكيلة (configuration) .

(٣) لا يوجد أي ملف إدخال خاص (special input file) . ونفرض أنه عند الوقت الابتدائي (initial time) يحتوي الشريط على بعض المحتويات المحددة . وبعض هذه المحتويات يمكن اعتبارها مدخلات (input) . وبالمثل لا توجد وحدة إخراج (output device) خاصة . وعندما تتوقف الآلة فإن بعض أو كل محتويات الشريط يمكن أن ننظر إليها على أنها مخرجات (output) .

وقد تم اختيار هذه الاصطلاحات (conventions) السابقة أساسا لملاءمة المناقشة التالية . وفي الفصل التالي سنرى بإذن الله صيغا أخرى لآلات تيورنج ، ونناقش علاقتها بنموذجنا القياسي .

وهنا – كما في حالة الآلات pda – فإن أنسب طريقة لعرض متتابعة تشكيلات (sequence of configurations) آلة تيورنج تستخدم فكرة الوصف اللحظي (instantaneous description) . وأي تشكيلة يتم تحديدها تماما بالحالة الحالية (current

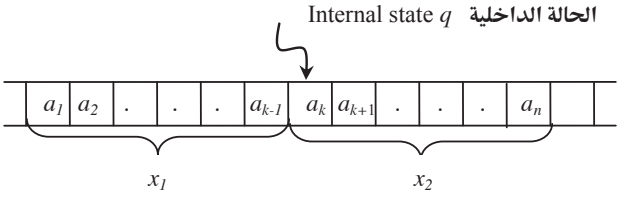
(state لوحدة التحكم ، ومحتويات الشريط ، وموضع رأس القراءة والكتابة . وسنستخدم بإذن الله الاصطلاح الذي يكون فيه

$$x_1 q x_2$$

أو

$$a_1\ a_2\ \ldots a_{k-1}\ q a_k\ a_{k+1}\ \ldots a_n$$

هو الوصف اللحظي لآلة في الحالة **q** حيث الشريط يوضحه الشكل التالي (شكل ١ – ٧) :



شكل ١ – ٧

الرموز $a_1, \ldots, a_n$ تبين محتويات الشريط ، بينما q تُعَرِّف حالة وحدة التحكم . وقد اخترنا هذا الاصطلاح كي يصبح موضع رأس القراءة والكتابة فوق الخلية التي تحتوي على الرمز الذي يلي q مباشرة .

والوصف اللحظي يعطي فقط قدرا محدودا من المعلومات يمين ويسار رأس القراءة والكتابة . وأما الجزء غير المحدد (unspecified part) من الشريط فإنه يُفترض أن يحتوي كلية على فراغات (all blanks) . وعادة ما تكون مثل هذه الفراغات لا أهمية لها ، ولا يتم عرضها صراحة في الوصف اللحظي . فمثلا الوصف اللحظي $w\ \square\ q$ يشير إلى أن رأس القراءة والكتابة تقع على الخلية الموجودة مباشرة يسار أول رمز من رموز w ، وأن هذه الخلية تحتوي على فراغ .

مثال ١ – ٥ :

الصور التي تظهر في شكل ١ – ٤ تقابل متتابعة الأوصاف اللحظية (sequence of instantaneous descriptions) :

$$q_0aa,\ bq_0a,\ bbq_0\square,\ bq_1b$$

* * *

وسنرمز للحركة (a move) من تشكيلةٍ (configuration) لأخرى بالرمز $\vdash$. فمثلا إذا كان

$$\delta(q_1, c) = (q_2, e, R)$$

فإن الحركة

$$abq_1cd \vdash abeq_2d$$

تتم كلما كانت الحالة الداخلية (internal state) هي q_1 ، والشريط يحتوي على ***abcd*** ، ورأس القراءة والكتابة (read-write head) واقفة عند c . وأما الرمز $\vdash^*$ فسيرمز كالمعتاد إلى عدد اختياري من التحركات . وسنستخدم بإذن الله مؤشرات / رموزاً سُفلى (subscripts) مثل $\vdash_M$ للتمييز بين عدة آلات .

مثال ١ – ٦ :

آلة تيورنج المبينة في شكل ١ – ٤ يمكننا تمثيل عملها (action) كما يلي :

$$q_0aa \vdash bq_0a + bbq_0\square \vdash bq_1b$$

أو كالتالي :

$$q_0aa \vdash^* bq_1b.$$

* * *

ولاستكمال دراستنا نرى أنه من المناسب أن نلخص الملاحظات السابقة في التعريف التالي :

تعريف ١ – ٣ : إذا كانت لدينا آلة تيورنج التالية

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$$

فإن :

(i) أي سلسلة $a_1 \dots a_{k-1}q_1a_ka_{k+1}\dots a_n$ حيث $a_i \in \Gamma$, $q_1 \in Q$ تُعَدُّ وصفاً لحظيا (an instantaneous description) للآلة M .

(ii) الحركة

$$a_1 \dots a_{k-1}q_1a_ka_{k+1}\dots a_n \vdash a_1 \dots a_{k-1}bq_2a_{k+1}\dots a_n$$

تكون ممكنة إذا وفقط إذا كانت

$$\delta(q_1, a_k) = (q_2, b, R)$$

(iii) الحركة

$$a_1 \ldots a_{k-1} q_1 a_k a_{k+1} \ldots a_n \vdash a_1 \ldots q_2 a_{k-1} b a_{k+1} \ldots a_n$$

تكون ممكنة إذا وفقط إذا كانت

$$\delta(q_1, a_k) = (q_2, b, L)$$

(iv) يقال إن الآلة M تتوقف (halt) مبتدئةً من تشكيلة ابتدائية (some initial configuration) $x_1 q_i x_2$ إذا كانت

$$x_1 q_i x_2 \vdash^* y_1 q_j a y_2$$

لأي q_j وأي a تكون عندهما $\delta(q_j, a)$ غير مُعَرَّفة (undefined).

ومتتابعة التشكيلات (sequence of configurations) التي تؤدي إلى حالة توقف (a halt state) يُطلق عليها حِسْبة / عملية حسابية (a computation).

يوضح مثال ١ – ٤ إمكانية عدم توقف آلة تيورنج إطلاقا (never halts)، حيث تدخل في عروة لا نهائية (endless loop) لا تستطيع الهروب منها. ونظرا لأن هذه الحالة تلعب دورا رئيسيا في دراسة عمل آلات تيورنج، فلذلك سنستخدم لها اصطلاحا خاصا، حيث نمثلها بالعلاقة

$$x_1 q x_2 \vdash^* \infty,$$

التي تشير إلى أنه إذا بدأنا بالتشكيلة الابتدائية $x_1 q x_2$ فإن الآلة ستدخل في عروة ولا تتوقف إطلاقا.

آلات تيورنج كآلات قبول للغة

Turing Machines as Language Accepters

يمكن أن ننظر لآلات تيورنج على أنها آلات قبول (accepters) بالمفهوم التالي: تُكتب أي سلسلة w (a string) على الشريط (tape)، مع مَلء (filling) الأجزاء غير المستخدَمة (unused portions) بفراغات (blanks)، ونبدأ الآلة بالحالة الابتدائية q_0، مع وضع رأس القراءة والكتابة على رمز السلسلة w الموجود أقصى يسارها (leftmost symbol of w). وبعد متتابعة من التحركات إذا دخلت الآلة حالة نهائية (a final state) وتوقفت، فإننا نعتبر أن السلسلة w قد تم قبولها.

تعريف ١ – ٤ : نفرض أن $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ **آلة تيورنج .**

اللغة التي تقبلها M **(the language accepted by) هي :**

$$L(M) = \{ w \in \Sigma^+ : q_0 w \vdash^* x_1 q_f x_2 \text{ for some } q_f \in F, x_1, x_2, \in \Gamma^* \}$$

هذا التعريف يشير إلى أن سلسلة الإدخال w **تُكتب على الشريط مع وجود الفراغات في أحد الجانبين . وأما السبب في استبعاد الفراغات من المدخلات فسيتضح الآن : إنه يؤكد لنا أن جميع المدخلات محصورة (restricted) في منطقة من الشريط مُعَرَّفة ومحددة تحديدا جيدا (well-defined region) ، محصورة بأقواس من الفراغات (bracketed by blanks) على اليمين واليسار . وبدون هذا الاصطلاح (convention) فإن الآلة لا تستطيع أن تضع حدا (limit) للمنطقة التي عليها أن تبحث فيها عن المدخلات . فبغض النظر عن عدد الفراغات التي تراها لا تستطيع أن تتأكد إطلاقا أنه لم توجد بعض المدخلات التي هي ليست فراغات (nonblank input) في مكان آخر على الشريط .**

والتعريف السابق (تعريف ١ – ٤) يخبرنا ماذا يجب أن يحدث عندما تكون $w \in L(M)$ **. ولكنه لا يذكر شيئا عن النتيجة (outcome) بالنسبة لأي مدخلات أخرى . وعندما لا تكون** w **في اللغة** $L(M)$ **فإنه يمكن أن يحدث أحد أمرين : أما أن تتوقف الآلة (halt) في حالة غير نهائية ، أو أن تدخل الآلة في عروة لا نهائية ولا تتوقف إطلاقا . وأي سلسلة لا تتوقف الآلة** M **بالنسبة لها تكون – بناءً على التعريف (by definition) – غير موجودة في اللغة** $L(M)$ **.**

مثال ١ – ٧ :

نفرض $\Sigma = \{0, 1\}$ **. صمِّم آلة تيورنج تقبل اللغة التي يرمز إليها التعبير المنتظم** 00^*.

الحل :

هذا تدريب سهل بإذن الله في برمجة آلات تيورنج . نبدأ عند الطرف الأيسر للمدخلات ، ونقرأ كل رمز ونتحقق أنه 0 . فإن كان صفرا فعلا فإننا نستمر بالتحرك لليمين . فإذا وصلنا إلى فراغ (a blank) دون أن يقابلنا أي شئ غير الصفر ، فإننا ننتهي (terminate) ونقبل السلسلة . أما إذا احتوت المدخلات على 1 في أي موضع ، فإن السلسلة ليست في $L(00^*)$ **، ونتوقف (halt) في حالة غير نهائية (nonfinal state) . ولتعقب الحسابات (keep track of**

(the computation) تكفينا حالتان داخليتان $Q = \{q_0, q_1\}$ (internal states) ، وحالة نهائية واحدة $F = \{q_1\}$ (final state) . وكدالة انتقال (transition function) يمكننا أن نأخذ

$$\delta(q_0, 0) = (q_0, 0, R),$$
$$\delta(q_0, \square) = (q_1, \square, R).$$

وطالما أن صفرا 0 يظهر تحت رأس القراءة والكتابة ، فإن الرأس ستتحرك إلى اليمين . فإذا ما قُرأ واحد 1 في أي لحظة فإن الآلة ستتوقف في الحالة غير النهائية q_0 ، نظرا لأن $\delta(q_0,1)$ غير مُعَرَّفة . ولاحظ أن آلة تيورنج تتوقف أيضا في حالة نهائية إذا بدأناها في الحالة q_0 على فراغ (a blank) . ويمكننا أن نفسر هذا على أنه قبول (acceptance) للرمز الخالي λ ، ولكن لأسباب فنية (technical) فإن السلسلة الخالية (empty string) لا يشملها تعريف ١ – ٤ .

* * *

ويُعَدُّ التعرف على لغات أكثر تعقيدا أكثر صعوبة . ونظرا لأن آلات تيورنج تحتوي على مجموعة تعليمات بدائية ، فإن الحسابات التي يمكننا أن نبرمجها بسهولة في لغة عالية المستوى (a higher level language) ، تكون عادة معقدة وغير بسيطة على آلة تيورنج . ومع ذلك فإن التعرف على هذه اللغات الأكثر تعقيدا لازال ممكنا ، ومن السهل فهم واستيعاب هذه المفاهيم ، كما توضح ذلك بإذن الله الأمثلة التالية .

مثال ١ – ٨ :

نفرض أن $\sum = \{a, b\}$. صمم آلة تيورنج تقبل اللغة

$$L = \{a^n b^n : n \geq 1\}$$

الحل :

سنحل المسألة بإذن الله بالطريقة التالية : نبدأ عند الرمز a الموجود أقصى اليسار ثم نستبعده بأن نحل محله رمزاً ما ، وليكن x . ثم نجعل رأس القراءة والكتابة تتحرك إلى اليمين لتجد الرمز b الموجود أقصى اليسار ، والذي نستبعده بدوره بأن نحل محله رمزا ما آخر ، وليكن y . وبعد ذلك نتحرك لليسار مرة أخرى إلى الرمز a الموجود أقصى اليسار ، ونستبدل به رمزا x ، ثم نتحرك إلى الرمز b الموجود أقصى اليسار ، ونستبدل به رمزا y ، وهكذا . وبالتحرك يسارا ويمينا بهذه الطريقة يمكننا أن نوائم (match) بين كل رمز a والرمز b المقابل له . وبعد فترةٍ ما إذا لم نجد متبقيا لدينا أي رمز a وأي رمز b ، فإن السلسلة يجب أن تكون في اللغة L .

وبدراسة تفاصيل الحل نصل إلى حل كامل فيه :

$$Q = \{q_0, q_1, q_2, q_3, , q_4\}, F = \{q_4\}, \Sigma = \{a, b\}, \Gamma = \{a, b, x, y, \square\}$$

ويمكن تقسيم الانتقالات (transitions) إلى عدة أجزاء :

(i) المجموعة الأولى :

$$\delta(q_0, a) = (q_1, x, R),$$
$$\delta(q_1, a) = (q_1, a, R),$$
$$\delta(q_1, y) = (q_1, y, R),$$
$$\delta(q_1, b) = (q_2, y, L)$$

هذه المجموعة تستبدل رمزا x بالرمز a الموجود أقصى اليسار ، ثم تجعل رأس القراءة والكتابة تتحرك يمينا إلى أول رمز b لتستبدل به رمزا y . وعندما يُكتب الرمز y ، فإن الآلة تدخل حالةً q_2 مشيرةً إلى أنه قد تمت بنجاح عملية مقابلة بين رمز a ورمزٍ مقابلٍ له b ، أي تمت مقابلة عنصري زوج a, b .

(ii) المجموعة الثانية : المجموعة التالية من الانتقالات تعكس الاتجاه (اتجاه حركة الآلة) حتى نقابل رمزاً x ، ثم تعيد وضع رأس القراءة والكتابة فوق الرمز a الموجود أقصى اليسار ، وتعيد التحكم (returns control) إلى الحالة الابتدائية:

$$\delta(q_2, y) = (q_2, y, L),$$
$$\delta(q_2, a) = (q_2, a, L),$$
$$\delta(q_2, x) = (q_0, x, R).$$

وقد عدنا الآن إلى الحالة الابتدائية q_0 ، مستعدين للتعامل مع الرمزين التاليين a, b .

بعد دورة واحدة [مرور واحد (one pass)] خلال هذا الجزء من الحسابات ، تكون الآلة قد أتمت إجراء الحسابات الجزئية (partial computation) التالية

$$q_0aa\ldots abb\ldots b \vdash^* xq_0a\ldots ayb\ldots b$$

حيث تمت المواءمة بين رمز واحد a ورمز واحد b .

وبعد دورتين (two passes) تكون الآلة قد أتمت إجراء الحسابات الجزئية التالية

$$q_0aa\ldots abb\ldots b \vdash^* xxq_0\ldots ayy\ldots b$$

وهكذا ، مشيرةً إلى أن عملية المواءمة يتم إجراؤها بطريقة صحيحة (properly) .

وعندما تكون المدخلات عبارة عن سلسلة $a^n\ b^n$ ، فإن عملية إعادة الكتابة (rewriting) تستمر بهذه الكيفية ، وتتوقف فقط حينما لا توجد أي رموز a أخرى ليتم مسحها (to be erased) . وعندما نبحث عن الرمز a الموجود أقصى اليسار فإن رأس القراءة والكتابة تتحرك يسارا حيث الآلة تكون في الحالة q_2 . وعندما نقابل رمزاً x ، فإنه يتم عكس الاتجاه لنصل إلى الرمز a . ولكننا الآن بدلا من أن نجد رمزا a فإننا سنجد رمزا y . وكي ننتهي فإننا نعمل اختبارا أخيرا (a final check) لنرى إن كانت جميع الرموز a و b قد تم إحلال رموز محلها [لنكتشف (detect) مدخلات يَتْبع فيها رمزٌ a رمزاً b] ويمكننا عمل ذلك الاختبار بالانتقالات التالية :

$$\delta\ (q_0, y) = (q_3, y, R),$$
$$\delta\ (q_3, y) = (q_3, y, R),$$
$$\delta\ (q_3, \square) = (q_4, \square, R).$$

مثال ١ – ٩ :

نفرض أن L هي اللغة المعطاة في المثال السابق (مثال ١ – ٨) وأن M هي آلة تيورنج التي قمت بتصميمها في هذا المثال .

أ) تَتَبَّع الأوصاف اللحظية المتعاقبة (successive instantaneous descriptions) بالنسبة لسلسلة المدخلات $aabb$ ، وتَبَيَّن قبول أو عدم قبول السلسلة .

ب) نفرض أننا قد أدخلنا للآلة M سلسلة المدخلات $a^n,\ b^m$ حيث $n > m$. في أي حالة ستتوقف الآلة ؟ وبالتالي تَتَبَّن قبول أو عدم قبول السلسلة .

الحل :

أ) سلسلة المدخلات $aabb$ تعطينا الأوصاف اللحظية المتعاقبة التالية :

$$q_0aabb \vdash xq_1abb \vdash xaq_1bb \vdash xq_2ayb$$
$$\vdash q_2xayb \vdash xq_0ayb \vdash xxq_1yb$$
$$\vdash xxyq_1b \vdash xxq_2yy \vdash xq_2xyy$$
$$\vdash xxq_0yy \vdash xxyq_3y \vdash xxyyq_3\square$$

$$\vdash xxyy\square q_4\square.$$

مما يعني أن الآلة ستتوقف (halts) في حالة نهائية "q_4" ، وبالتالي فإن السلسلة ***aabb*** ستُقبل .

ب) بتتبُّع الأوصاف اللحظية المتعاقبة سنرى أن الآلة ستقابل (encounter) أخيرا فراغا (a blank) في الحالة q_1 . وستتوقف الآلة نظرا لأنه لا يوجد انتقال (transition) محدد لهذا الوضع . أي أن الآلة توقفت عند حالة غير نهائية (nonfinal halting state) . وبالتالي فإن سلسلة المدخلات المعطاة لا تُقبل ، أي غير موجودة في اللغة L .

ويمكن للقارئ الكريم أن يتتبع برنامج مثال ١ – ٨ بعدة سلاسل مدخلات مختلفة ، بعضها في اللغة L وبعضها ليس في اللغة L .

مثال ١ – ١٠ :

صمم آلة تيورنج تقبل اللغة

$$L = \{a^n b^n c^n : n \geq 1\}$$

الحل :

الأفكار التي استخدمناها في حل مثال ١ – ٨ يمكن أن نطبقها بسهولة بإذن الله في مثالنا الحالي . فنوائم (match) كل ثلاثة رموز a, b, c بأن نستبدل بها على الترتيب الرموز الثلاثة x, y, z . وفي النهاية نتحقق من أن جميع الرموز الأصلية (original symbols) قد تم إحلال رموز محلها [قد تمت إعادة كتابتها (rewritten)] . ورغم أن مفهوم امتداد (extension) حل مثال ١ – ٨ يبدو بسيطا ، إلا أن كتابة البرنامج الفعلي ستكون مملة وطويلة ولكنها مباشرة . ولاحظ أنه رغم أن $\{a^n b^n\}$ لغة حرة السياق (a context-free language) بينما $\{a^n b^n c^n\}$ ليست لغة حرة السياق ، إلا أنه يمكن قبولهما بآلات تيورنج ببنْيات متشابهة جدا (very similar structures) .

* * *

أحد الاستنتاجات التي نصل إليها من هذا المثال الأخير هو أن آلة تيورنج يمكن أن تتعرف على بعض اللغات غير حرة السياق ، مما يدل دلالة أولية على أن آلات تيورنج أقوى (more powerful) من آلات الدفع لأسفل (pushdown automata) .

آلات تيورنج كمحولات للطاقة Turing Machines as Transducers

حتى الآن لم يكن لدينا أي دافع قوي لدراسة محولات الطاقة (transducers) . ففي نظرية اللغات (language theory) تُعد آلات القبول (accepters) مناسبة جدا وكافية . ولكن كما سنرى بإذن الله بعد قليل فإن آلات تيورنج لا تفيدنا فقط كآلات قبول للغات (language accepters) ، ولكنها تعطينا أيضا نموذجا مجردا بسيطا (a simple abstract model) للحواسيب الرقمية (digital computers) عموما . ونظرا لأن الغرض الأساسي من الحاسوب هو تحويل (transform) المدخلات إلى مخرجات ، فإنه يعمل كمحول للطاقة . فإذا أردنا أن نضع نماذج (models) للحواسيب باستخدام آلات تيورنج ، فعلينا أن ننظر إلى هذا الموضوع أكثر عمقا .

مدخلات أي عملية حسابية (a computation) ستكون هي جميع الرموز غير الفراغات (nonblank symbols) والموجودة على الشريط عند بداية الوقت (at the initial time) . وعند نهاية الحسابات (conclusion of the computation) فإن المخرجات ستكون هي ما هو موجود وقتها على الشريط . وهكذا فإنه يمكننا أن ننظر إلى آلة تيورنج المحولة للطاقة (a Turing machine transducer) $\boldsymbol{M}$ على أنها تنفيذ (implementation) لدالة $\boldsymbol{f}$ مُعَرَّفة كما يلي :

$$\hat{w} = f(w)$$

بشرط أن

$$q_0 w \vdash^*_M q_f \hat{w}$$

لحالةٍ نهائيةٍ ما q_f .

تعريف ١ – ٥ :

يُقال لدالة $\boldsymbol{f}$ مجالها $\boldsymbol{D}$ (domain) إنها قابلة للحساب على آلة تيورنج (Turing computable) - ، أو ببساطة قابلة للحساب (computable) إذا وُجدت آلة تيورنج ما $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ بحيث أن

$$q_0 w \vdash^*_M q_f f(\mathrm{w}), \quad q_f \in F, \quad \forall\ w \in D$$

وكما سنزعم (claim) بعد قليل بإذن الله فإن جميع الدوال الرياضية الشائعة (common mathematical functions) – مهما كانت درجة تعقيدها – قابلة للحساب على

آلة تيورنج . ونبدأ بعون الله بالنظر إلى بعض العمليات البسيطة كالجمع (addition) ، والمقارنة الحسابية (arithmetic comparison) .

مثال ١ – ١١ :

نفرض أن x, y عددان صحيحان موجبان . صمم آلة تيورنج تحسب المجموع $x + y$.

الحل :

علينا أولا أن نختار اصطلاحاً ما لتمثيل الأعداد الصحيحة الموجبة . وللبساطة سنستخدم الاصطلاح الأحادي (unary notation) الذي يُمثَّل فيه أي عدد صحيح موجب x هكذا $w(x) \in \{1\}^+$ بحيث أن

$$|w(x)| = x$$

كما يجب علينا أن نقرر كيف سنضع العددين x, y على الشريط في البداية (initially) ، وكيف سيظهر مجموعهما في نهاية الحسابات . نفرض أن $w(x), w(y)$ سيوضعان على الشريط بالاصطلاح الأحادي ، يفصل بينهما صفر مفرد 0 (a single) ، مع وجود رأس القراءة والكتابة على رمز $w(x)$ الموجود أقصى اليسار . وبعد إجراء الحسابات سيكون $w(x + y)$ على الشريط يعقبه صفر مفرد 0 (a single) ، وستكون رأس القراءة والكتابة عند النهاية اليسرى (left end) للنتيجة . ولذلك فإننا نود تصميم آلة تيورنج تقوم بإنجاز الحسابات

$$q_0 w(x)\, 0 w(y) \vdash^* q_f w(x+y)\, 0$$

حيث q_f حالة نهائية . وعملية إنشاء برنامج (constructing a program) لهذا الغرض تُعَدُّ بسيطة نسبيا . فكل ما نحتاج عمله هو أن نُحَرِّك الصفر 0 الفاصل (separating 0) إلى نهاية $w(y)$ اليمنى ، بحيث أن عملية الجمع لا تصبح أكثر من مجرد عملية دمج (coalescing) السلسلتين . ولتحقيق ذلك ننشئ آلة

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$$

حيث

$$Q = \{q_0, q_1, q_2, q_3, , q_4\}, F = \{q_4\},$$

$$\begin{aligned}
&\delta(q_0, 1) = (q_0, 1, R),\\
&\delta(q_0, 0) = (q_1, 1, R),\\
&\delta(q_1, 1) = (q_1, 1, R),\\
&\delta(q_1, \square) = (q_2, \square, L),
\end{aligned}$$

$$\delta(q_2, 1) = (q_3, 0, L),$$
$$\delta(q_3, 1) = (q_3, 1, L),$$
$$\delta(q_3, \square) = (q_4, \square, R).$$

لاحظ أننا بتحريك الصفر 0 إلى اليمين فإننا نُنشئ مؤقتا واحدا 1 إضافيا (create an extra 1) ، وهي حقيقة نتذكرها (remember) بوضع الآلة في الحالة q_1 . والانتقال $\delta(q_2, 1) = (q_3, 0, R)$ نحتاج إليه لإزالة هذا (الواحد الإضافي) في نهاية الحسابات . والمثال التالي يوضح هذه العملية .

مثال ١ – ١٢ :

استخدم تصميم آلة تيورنج المذكور في حل مثال ١ – ١١ لجمع العددين الصحيحين الموجبين (بالاصطلاح الأحادي) 11 ,111 مع بيان متتابعة الأوصاف اللحظية لعملية الجمع .

الحل :

$$q_0111011 \vdash 1q_011011 \vdash 11q_01011 \vdash 111q_0011$$
$$\vdash 1111q_111 \vdash 11111q_11 \vdash 111111q_1\square$$
$$\vdash 11111q_21 \vdash 1111q_310$$
$$\vdash^* q_3\square111110 \vdash q_4111110.$$

ملاحظة :

رغم أن استخدام الاصطلاح الأحادي ممل وطويل بالنسبة للحسابات العملية ، إلا أنه ملائم جدا لبرمجة آلات تيورنج . فالبرامج الناتجة أقصر بكثير وأبسط من تلك التي نتوصل إليها عند استخدام طرق تمثيل أخرى كالتمثيل الثنائي (binary) ، أو التمثيل العشري (decimal) .

* * *

تُعَدُّ إضافة / جمع الأعداد إحدى العمليات الأساسية بالنسبة لأي حاسوب ، والتي تلعب دورا هاما في تركيب (synthesis) تعليمات (instructions) أكثر تعقيدا . والعمليات الأساسية الأخرى هي نسخ السلاسل (copying strings) والمقارنات البسيطة (simple comparisons) . وهذه العمليات يمكن أيضا إجراؤها بسهولة باستخدام آلة تيورنج .

مثال ١ – ١٣ :

صمم آلة تيورنج تنسخ سلاسل من الآحاد 1's (copies strings of) **. وبتعبير أدق: أوجد آلة تقوم بإجراء الحسابات**

$$q_0 w \vdash^* q_f ww$$

وذلك لأي

$$w \in \{1\}^+$$

وارسم مخطط الانتقالات البياني لهذه الآلة .

الحل :

لحل هذه المسألة نقوم بتنفيذ الخطوات البديهية التالية :

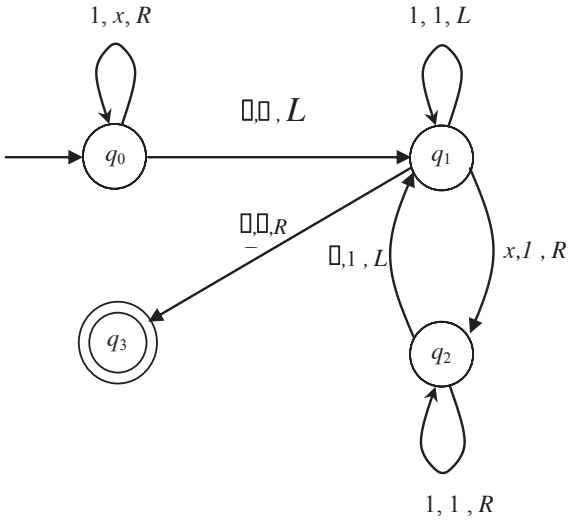
(١) استبدل بكل واحد 1 **رمزاً** x **.**

(٢) أوجد الرمز x **الموجود أقصى اليمين ، واستبدل به واحدا** 1**.**

(٣) تحرك إلى النهاية اليمنى (right end) **للمنطقة الحالية للرموز غير الفراغية** (current nonblank region) **وأنشئ واحداً** 1 (create a) **هناك .**

(٤) كرر الخطوتين (٢) و (٣) إلى أن لا يبقى لدينا أي رمز x **.**

وهذا الحل مبين في مخطط الانتقالات البياني التالي (شكل ١ – ٨) .



شكل ١– ٨

قد يبدو من الصعب في البداية التحقق من صحة هذا الحل ، ولكن المثال التالي يوضح بإذن الله صحة هذا الحل عن طريق تتبع خطواته .

مثال ١ – ١٤ :

تتبع (trace) برنامج المثال السابق (مثال ١ – ١٣) باستخدام متسلسلة المدخلات البسيطة 11 ، وبيان الحسابات المتعاقبة التي يتم إجراؤها .

الحل :

$$\begin{aligned} q_0 11 &\vdash x q_0 1 \vdash x x q_0 \square \vdash x q_1 x \\ &\vdash x 1 q_2 \square \vdash x q_1 11 \vdash q_1 x 11 \\ &\vdash 1 q_2 11 \vdash 11 q_2 1 \vdash 111 q_2 \square \\ &\vdash 11 q_1 11 \vdash 1 q_1 111 \\ &\vdash q_1 1111 \vdash q_1 \square 1111 \vdash q_3 1111. \end{aligned}$$

مثال ١ – ١٥ :

نفرض أن x, y عددان صحيحان موجبان مُمَثَّلان بالاصطلاح الأحادي . أنشئ آلة تيورنج تتوقف (halt) في حالة نهائية q_y إذا كانت $x \geq y$ ، وتتوقف في حالة غير نهائية q_n إذا كانت $x < y$. وبصيغة أكثر تحديدا فإن الآلة تقوم بإجراء الحسابات التالية :

$$q_0 w(x)\, 0 w(y) \vdash^* q_y w(x)\, 0 w(y) \qquad \text{if } x \geq y,$$

$$q_0 w(x)\, 0 w(y) \vdash^* q_n w(x)\, 0 w(y) \qquad \text{if } x < y.$$

الحل :

لحل هذه المسألة يمكننا استخدام فكرة حل مثال ١ – ٨ مع بعض التعديلات البسيطة . فبدلا من المواءمة بين الرموز a والرموز b ، نوائم بين كل واحد 1 على يسار الصفر 0 الفاصل (dividing 0) والواحد 1 على اليمين . وفي نهاية المواءمة سيكون لدينا على الشريط إما

$$xx\ldots 110xx\ldots x\square$$

أو

$$xx\ldots xx0xx\ldots x11\square$$

وذلك حسب ما إذا كانت $x > y$ أو $y > x$. وفي الحالة الأولى حينما نحاول أن نوائم واحداً 1 آخر فسيقابلنا الفراغ (blank) الموجود على يمين الحيِّز الذي نعمل فيه (working space) . وهذا يمكن أن يُستخدم كإشارة (signal) للدخول في الحالة q_y . وأما في الحالة الثانية فسنظل نجد واحداً 1 على اليمين عندما تكون جميع الآحاد 1's على اليسار قد تم إحلال رموز محلها . ونستخدم ذلك للدخول في الحالة الأخرى q_n . والبرنامج الكامل لهذه المسألة يُعَدُّ مباشرا ونتركه كتدريب للقارئ الكريم .

وهذا المثال يوضح نقطة هامة وهي أن آلة تيورنج يمكن أن تُبرمج لاتخاذ قرارات (making decisions) بناءً على مقارنات حسابية . وهذا النوع من اتخاذ قرارات بسيطة شائع في لغة الآلة (machine language) في الحواسيب ، حيث يتم إدخال سيل من التعليمات البدائل (alternate instruction streams) بناءً على نتيجة (outcome) عملية حسابية (arithmetic operation) .

الجمع بين آلات تيورنج لتنفيذ مهام معقدة
Combining Turing Machines for Complicated Tasks

رأينا صراحةً كيف أن بعض العمليات الهامة الموجودة في جميع الحواسيب يمكن أن ننفذها بواسطة آلة تيورنج . ونظرا لأنه في الحواسيب الرقمية تكون هذه العمليات الأساسية / البدائية قوالب البناء (building blocks) لتعليمات أكثر تعقيدا ، فسنرى بإذن الله تعالى كيف يمكننا أيضا وضع هذه العمليات الأساسية مع بعضها البعض (put together) على آلة تيورنج . ولعرض وتوضيح كيفية الجمع بين آلات تيورنج سنتبع أسلوبا شائعاً في البرمجة ، حيث نبدأ بوصف عالي المستوى (high-level description) ثم نقوم بصقله وتفصيله وتنقيحه (refining) على التتابع حتى يصبح البرنامج باللغة الفعلية التي نعمل بها . ويمكننا أن نصف آلات تيورنج بعدة طرق عالية المستوى وفي معظم المرات سنستخدم إما المخططات البيانية (القالبية) (block diagrams) أو الكود / البرامج الشبيهة (الزائفة) (pseudo codes) في مناقشاتنا التالية . وفي حالة المخططات القالبية فإننا نغلِّف (encapsule) الحسابات في قوالب / صناديق (boxes) توصَف مهماتها ، ولكن تفاصيلها الداخلية لا تُظهر . وباستخدام هذه القوالب نزعم ضمنيا أنه يمكن فعليا إنشاؤها وتركيبها . وكمثال أولي لتوضيح ما سبق نجمع بين آلتي مثالي ١ – ١١ و ١ – ١٥ .

مثال ١ – ١٦ :

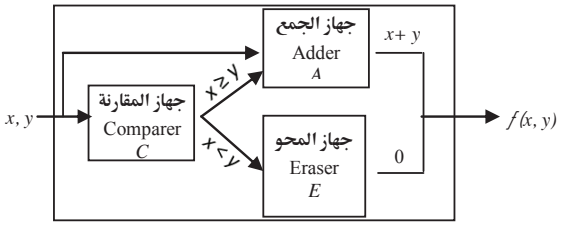
صمم آلة تيورنج تحسب قيمة الدالة

$$f(x, y) = x + y \qquad \text{if } x \geq y.$$
$$= 0 \qquad \text{if } x < y.$$

الحل :

لسهولة المناقشة سنفرض أن x, y عددان صحيحان موجبان ممثَّلان بالصيغة الأحادية . والقيمة صفر ستُمثَّل بالرمز 0 مع بقاء بقية الشريط فراغا (blank) .

يمكننا أن نصور عملية حساب قيمة الدالة $f(x, y)$ على مستوى عالٍ بواسطة المخطط المبين في شكل ١ – ٩ . والمخطط يبين أننا نستخدم أولا آلة مقارنة (a comparing machine) كآلة مثال ١ – ١٥ لتحديد ما إذا كانت $x \geq y$ أم لا . فإن كانت $x \geq y$ فإن جهاز



شكل ١ – ٩

المقارنة (comparer) يرسل إشارة بدء (a start signal) إلى جهاز الجمع (adder) الذي يقوم عندئذ بحساب المجموع $x + y$. وإن كانت $x < y$ فإننا نبدأ (start) برنامج محو (an erasing program) ليقوم بتغيير كل 1 إلى فراغ (a blank) .

* * *

في مناقشاتنا التالية سنستخدم غالبا مثل هذا التمثيل (representation) عالي المستوى بالمخططات البيانية القالبية لآلات تيورنج . وهذه الطريقة لا شك أسرع وأوضح من طريقة استخدام مجموعة الانتقالات δ's الموسَّعة المقابلة (the corresponding extensive set of δ's) . وقبل أن نقبل هذه النظرة عالية المستوى يجب أن نبررها . فمثلا ماذا نعني بقولنا إن جهاز المقارنة يرسل إشارة بدء إلى جهاز الجمع ؟ فلا يوجد شئ في تعريف ١ – ١ يعرض هذه الإمكانية . ومع ذلك فيمكن تنفيذ ذلك بطريقة مباشرة .

أما برنامج جهاز المقارنة **C** فنكتبه كما اقترحناه في مثال ١ – ١٥ ، مستخدمين آلة تيورنج تحتوي على حالات مرتبطة / مرقَّمة / مؤشرة (indexed with) بجهاز المقارنة **C** . وأما بالنسبة لجهاز الجمع فنستخدم فكرة مثال ١ – ١١ مع جعل الحالات مؤشَّرة بجهاز الجمع **A** . وأما بالنسبة لجهاز المحـــو **E** فننشئ آلة تيـورنج تحتـوي على حـالات مؤشَّرة بجهـاز المحو **E** . والحسابات (computations) التي على جهاز المقارنة أن يقوم بها هي :

$$q_{C,0}w\,(x)\;0w\,(y)\;\vdash^{*} q_{A,0}w\,(x)\;0w\,(y)\quad \text{if } x \geq y,$$

وكذلك

$$q_{C,0}w\,(x)\;0w\,(y)\;\vdash^{*} q_{E,0}w\,(x)\;0w\,(y)\quad \text{if } x < y$$

فإذا أخـذنا $q_{A,0}$ على أنها الحالة الابتدائية لجهاز الجمع A ، وكذلك $q_{E,0}$ على أنها الحالة الابتدائية لجهاز المحو E ، فنرى أن جهاز المقارنة C سَيُبْدِئ (starts) إما جهاز الجمع A أو جهاز المحو E .

والحسابات التي سيقوم جهاز الجمع بإنجازها هي :

$$q_{A,0}w\,(x)\;0w\,(y)\;\vdash^{*} q_{A,f}\,w\,(x+y)\;0$$

والحسابات التي سيقوم جهاز المحو بتنفيذها هي :

$$q_{E,0}w\,(x)\;0w\,(y)\;\vdash^{*} q_{E,f}\,0$$

والنتيجة هي آلة تيورنج واحدة تجمع بين عمل كل من جهاز مقارنة C ، وجهاز جمع A ، وجهاز محو E كما هو مبين في شكل ١ – ٩ .

* * *

وهناك نظرة (view) أخرى مفيدة وعالية المستوى بالنسبة لآلات تيورنج تشتمل على كـود شبيه / زائف (pseudocode) . وفي برمجة الحاسوب يُعَدُّ الكـود الزائف طريقة جيدة لتلخيص وتحديد الإطار العام للحسابات باستخدام عبارات وصفية (descriptive phrases) يسهل فهم معناها . وبينما لا يكون هذا الوصف قابلا للاستخدام على الحاسوب ، إلا أننا نفترض أنه يمكن ترجمته إلى اللغة المناسبة حين نحتاج إلى ذلك . وأحد الأمثلة للأنواع البسيطة للكـود الزائف فكـرة التعليمـة الماكـرو / الإجماليـة (a macroinstruction) ، والتي هي عبـارة عـن عبـارة مفـردة (a single statement) تُعَدُّ صيغـة مختـصرة لمتتابعـة من عبارات منخفضـة

المستوى (a sequence of lower - level statements) . وسنُعَرِّف أولا التعليمة الماكرو بدلالة اللغة الأقل مستوى . ثم نستخدم التعليمة الماكرو في برنامج بفرض أن الكود الأقل مستوى ذا العلاقة (relevant low - level code) سيحل محل (substitutes for) التعليمة الماكرو كلما ظهرت . وهذه الفكرة مفيدة جدا في برمجة آلات تيورنج .

مثال ١ – ١٧ :

نفرض أن لدينا التعليمة الماكرو

$$\text{if } a \text{ then } q_j \text{ else } q_k$$

وتفسيرها كالتالي

- إذا قرأت آلة تيورنج رمزاً a ، فإنها بغض النظر عن حالتها الحالية (current state) تنتقل إلى الحالة q_j بدون تغيير محتويات الشريط أو تحريك رأس القراءة والكتابة .
- وإذا لم يكن الرمز المقروء a ، فإن الآلة تنقل إلى الحالة q_k بدون تغيير أي شئ .

وتنفيذ هذه التعليمة الماكرو يتطلب عدة خطوات واضحة نسبيا من آلة تيورنج :

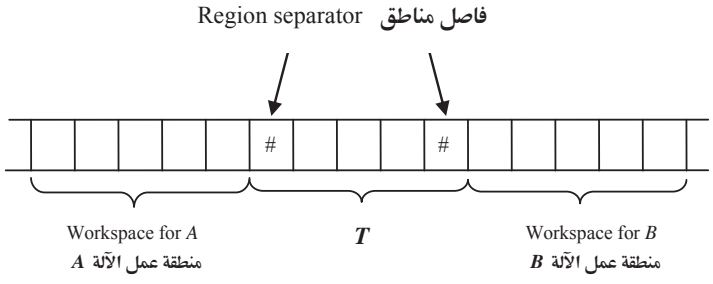
$$\begin{aligned}
&\delta(q_i, a) = (q_{j0}, a, R) \quad \forall\ q_i \in Q,\\
&\delta(q_i, b) = (q_{k0}, b, R) \quad \forall\ q_i \in Q,\ \forall\ b \in \Gamma - \{a\},\\
&\delta(q_{j0}, c) = (q_j, c, L) \quad \forall\ c \in \Gamma,\\
&\delta(q_{k0}, c) = (q_k, c, L) \quad \forall\ c \in \Gamma,
\end{aligned}$$

الحالتان q_{j0} ، q_{k0} حالتان جديدتان تم تعريفهما لتأخذا في الاعتبار وتعالجا التعقيدات الناشئة عن حقيقة أنه في آلة تيورنج القياسية تُغَيِّر رأس القراءة والكتابة موضعها في كل حركة (move) ، بينما في التعليمة الماكرو نود أن نغيِّر الحالة ولكن نترك رأس القراءة والكتابة في موضعها . فنجعل الرأس تتحرك لليمين ، ولكن نضع الآلة في حالة q_{j0} أو q_{k0} . وهذا يشير إلى أنه يجب عمل حركة لليسار قبل الدخول في الحالة المطلوبة q_j أو q_k .

* * *

وإذا تقدمنا الآن خطوة أبعد فيمكننا أن نستبدل بتعليمات الماكرو برامج فرعية (subprograms) . وعادة فإننا نستبدل بأي تعليمة ماكرو الكود الفعلي عند كل ظهور للتعليمة ، بينما أي برنامج فرعي عبارة عن قطعة كود واحدة نستدعيها تكراريا كلما احتجنا إليها . والبرامج

الفرعية تُعد أساسية بالنسبة للغات البرمجة عالية المستوى ، ولكن يمكننا أيضا استخدامها مع آلات تيورنج . ولتَقَبُّل ذلك دعنا نحدد باختصار الخطوط العريضة لكيفية استخدام آلة تيورنج كبرنامج فرعي يمكن استدعاؤه تكراريا بواسطة آلة تيورنج أخرى . وهذا يتطلب خاصية جديدة وهي القدرة على تخزين معلومات عن تشكيلة البرنامج المستدعِي (calling program's configuration) حتى يمكن إعادة إنشاء (recreating) التشكيلة عند العودة (return) من البرنامج الفرعي . فمثلا إذا قلنا إن الآلة A في الحالة q_i تستدعِي الآلة B . عندما ننتهي من B ، نود أن نستأنف برنامج A في الحالة q_i ، وحيث رأس القراءة والكتابة (والتي ربما تكون قد تحركت أثناء عمل B) تكون في موضعها الأصلي . وفي أوقات أخرى قد تَستدعي الآلة A الآلة B من الحالة q_j ، وعندئذ يجب أن يعود التحكم (control) إلى هذه الحالة . ولحل مشكلة نقل التحكم (control transfer problem) يجب أن نكون قادرين على تمرير المعلومات من A إلى B والعكس ، وأن نكون قادرين على إعادة إنشاء تشكيلة A عندما تستعيد التحكم من B ، وأن نضمن أن حسابات A التي تم توقيفها مؤقتا (temporarily suspended) لم تتأثر بتنفيذ B . ولحل هذه المشكلة يمكننا تقسيم الشريط إلى عدة مناطق كما هو مبين في شكل ١ – ١٠ .



شكل ١ – ١٠

قبل أن تَستدعي الآلة A الآلة B تقوم A بكتابة المعلومات التي تحتاجها B [مثلا : الحالة الحالية للآلة A ، والوسطاء للآلة B (arguments for)] على الشريط في منطقةٍ ما T . ثم تُمَرِّر A التحكم إلى B (passes control to) بتنفيذ انتقالٍ إلى الحالة الابتدائية في B . وبعد الانتقال تستخدم B المنطقة T لتجد مدخلاتها . ومنطقة عمل الآلة B منفصلة عن T وعن منطقة عمل الآلة A ، وبالتالي لا يمكن أن يحدث أي تداخل . وعندما تنتهي B من عملها ستعيد النتائج المطلوبة إلى المنطقة T ، حيث تتوقع A أن تجدها . وبهذه الطريقة يمكن أن يتفاعل (interact) البرنامجان بالأسلوب المطلوب . ولاحظ أن هذا شبيه جدا لما يحدث فعلا في حاسوب حقيقي عندما يتم استدعاء برنامج فرعي .

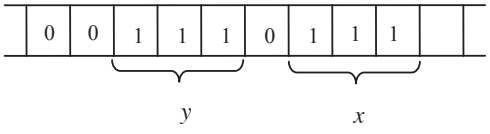
ويمكننا الآن أن نبرمج آلات تيورنج بكود شبيه (pseudocode) بشرط أن نعرف (نظريا على الأقل) كيف نترجم هذا الكود الشبيه إلى برنامج آلة تيورنج فعلي .

مثال ١ –١٨ :

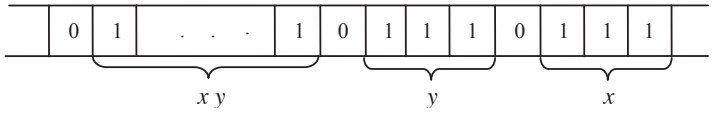
صمم آلة تيورنج تضرب عددين صحيحين موجبين في الاصطلاح الأحادي .

الحل :

يمكننا إنشاء آلة ضرب (a multiplication machine) بالجمع بين الأفكار التي مرت معنا في كل عمليتي الجمع (adding) والنسخ (copying) . دعنا نفرض أن محتويات الشريط الابتدائية ومحتويات الشريط النهائية تبدو كما هو مبين في شكل ١ – ١١ .



محتويات الشريط الابتدائية



محتويات الشريط النهائية

شكل ١ – ١١

يمكننا عندئذ أن ننظر إلى عملية الضرب على أنها عملية نسخ مكرر (a repeated copying) للعدد المضروب فيه **y**(multiplicand) وذلك لكل 1 في العدد الضارب **x** (multiplier) ، وحيث تُضاف (added) السلسلة **y** العدد المناسب من المرات إلى حاصل الضرب الجزئي الذي تم حسابه (partially computed product) . وفيما يلي الكود الشبيه الذي يبيِّن الخطوات الرئيسية لعملية الضرب

(١) كرر الخطوتين التاليتين إلى أن تصبح **x** خالية من أي آحاد **1's** :

(i) ابحث عن واحد **1** في **x** ، واستبدل به رمزا آخر a .

(ii) استبدل بالصفر **0** الموجود أقصى اليسار $\mathbf{0}_y$.

(٢) استبدل آحادا **1's** بجميع الرموز a**'s** .

ورغم أن هذا الكود الشبيه مجرد مخطط إلا أن الفكرة بسيطة جدا بحيث أنه يمكن بإذن الله تنفيذها دون أدنى شك .

* * *

رغم الطبيعة الوصفية (descriptive nature) للأمثلة السابقة ، إلا أنه ليس من المستبعد أن نستنتج منها تصوُّر أن آلات تيورنج – رغم أنها بدائية (primitive) من حيث المبدأ – يمكن الجمع بينها (combined) بعدة طرق لنجعلها ذوات قدرة عالية (quite powerful) . وأمثلتنا السابقة لم تكن عامة (general) ومفصلة (detailed) تفصيلا كافيا للادعاء بأننا قد أثبتنا شيئا ، ولكنها توحي إلينا في هذه المرحلة أن آلات تيورنج يمكنها أن تقوم بتنفيذ بعض المهام المعقدة .

رسالة تيورنج Turing's Thesis

تَبَيَّن لنا من المناقشة السابقة كيف يمكن إنشاء آلة تيورنج من أجزاء أبسط ، ولكن تَبَيَّن لنا أيضا في الوقت نفسه جانب سلبي للعمل بمثل هذه الآلات منخفضة المستوى (low-level automata) . فمع أن ترجمة مخطط قالبي أو كود شبيه إلى برنامج آلة تيورنج مقابل يستغرق منا تخيلا وتفكيرا بسيطا ، إلا أن تنفيذ ذلك فعليا يستغرق وقتا طويلا ويكون عرضة للخطأ . ومجموعة تعليمات (instruction set) آلة تيورنج تُعَدُّ محدودة جدا (so restricted) حتى أن أي حل أو نقاش أو برهان لمسألة غير بسيطة (nontrivial problem) يُعَدُّ طويلا جدا ومملا .

والآن تواجهنا مشكلة : فنحن نود أن نزعم أن آلات تيورنج يمكنها أن تقوم بتنفيذ ليس فقط العمليات البسيطة التي عرضنا لها برامج صريحة ، وإنما أيضا عمليات أكثر تعقيدا يمكن وصفها بمخططات قالبيه أو بكود شبيه . ولإثبات هذا الزعم علينا أن نبيِّن البرامج المقابلة لذلك صراحةً . ولكن ذلك أمر غير يسير وممل وطويل ومزعج ونود تجنبه ما أمكننا . ولذلك نبحث عن حل وسطي توافقي معقول ومقبول (reasonable compromise) . وفيما يلي نحاول الوصول إلى هذا الحل التوافقي .

يمكننا الوصول إلى عدة استنتاجات بسيطة من أمثلتنا السابقة . أول هذه الاستنتاجات هو أن آلات تيورنج تبدو وكأنها أقوى وأكثر قدرة من آلات الدفع لأسفل [ولتعليق على هذا الاستنتاج انظر المسألة ١ – ٩ بنهاية هذا الفصل] . وفي مثال ١ – ١٠ وضعنا مخططا (a sketch) لإنشاء (construction) آلة تيورنج للغةٍ ليست حرة السياق ، وبالتالي لا توجد لها آلة دفع لأسفل . وبَيَّنَت لنا الأمثلة ١ – ١١ و ١ – ١٣ و ١ – ١٥ أن آلات تيورنج يمكنها أن تقوم بإجراء بعض العمليات الحسابية البسيطة ، وبعض المعالجات للسلاسل (string manipulations) ، وبعض المقارنات . كما بَيَّنَت لنا المناقشة كيف يمكننا الجمع بين عمليات بدائية (primitive operations) لحل مسائل أكثر تعقيدا ، وكيف يمكننا تركيب / تكوين (composing) عدة آلات تيورنج ، وكيف يمكن لبرنامج أن يعمل كبرنامج فرعي (subprogram) لبرنامج آخر . ونظراً لأنه يمكن بناء (building) عمليات معقدة جدا بهذه الطريقة ، فقد نشك أن آلة تيورنج قد بدأت الاقتراب من الحاسوب المعتاد من حيث القدرة (power) .

هل يمكننا إذن أن نصل إلى التخمين (conjecture) / الفرض (hypothesis) أن: آلات تيورنج مساوية في القدرة (equal in power) لحاسوب رقمي نمطي (a typical digital computer) ؟

(*) لتأييد هذا الفرض : يمكننا أن نأخذ عدة مسائل أكثر صعوبة ونبيِّن كيف يمكن حلها باستخدام آلات تيورنج . ويمكننا كذلك أن نأخذ مجموعة تعليمات لغة الآلة الخاصة بحاسوب معين ونصمم آلة تيورنج يمكنها أن تنفذ جميع التعليمات في المجموعة . ورغم أن كل نجاح في هذا الاتجاه يقوي من اقتناعنا بصحة الفرض ، إلا أنه لا يؤدي إلى برهان . والصعوبة تكمن في أننا لا نعرف بالضبط ما المقصود بـ " حاسوب رقمي نمطي " ، ولا توجد لدينا وسيلة لتحديد تعريف دقيق .

ومن ناحية أخرى :

(*) لدحض هذا الفرض : يمكننا أن نحاول أن نجد إجراءً (procedure) ما يمكننا أن نكتب له برنامج حاسوب (a computer program) ، ولكن يمكننا إثبات أنه لا يمكن أن توجد له آلة تيورنج . فإن كان ذلك ممكنا فسيكون لدينا أساس لرفض هذا الفرض . ولكن حتى الآن لم يستطع أحد أن يعطي مثالا مناقضا (a counterexample) . وحقيقة فشل جميع مثل هذه المحاولات يجب أن تُؤخذ على أنها دلالة على أو إشارة إلى استحالة هذه الإمكانية . وجميع الدلالات تشير إلى أن آلات تيورنج من حيث المبدأ قوية وقادرة (powerful) كقدرة أي حاسوب .

وقد أدت مناقشات من هذا النوع بتيورنج (A. M. Turing) وآخرين في منتصف الثلاثينيات (mid-1930s) إلى التخمين / الفرض المعروف باسم رسالة تيورنج (Turing thesis) . وينص هذا الفرض (hypothesis) على أن : أي حسابات يمكننا تنفيذها

بوسائل ميكانيكية (mechanical means) ، سيمكننا أيضا تنفيذها بواسطة آلة ما من آلات تيورنج .

وهذه العبارة (رسالة تيورنج) ليست شيئا يمكن برهنته . فللوصول إلى ذلك علينا أن نُعَرِّف بدقة الاصطلاح " وسائل ميكانيكية ". وهذا سيتطلب بعض النماذج المجردة (abstract models) الأخرى ، ولا يجعلنا نتقدم للأمام أبعد مما كنا عليه سابقا . ولذلك فإن رسالة تيورنج يُنظر إليها بصورة أنسب على أنها تعريف لما يكوِّن حِسبة / عملية حسابية ميكانيكية (a mechanical computation) : " أي عملية حسابية (a computation) تكون ميكانيكية (mechanical) إذا وفقط إذا أمكن تنفيذها بواسطة آلةٍ ما من آلات تيورنج " .

فإذا تَبَنَّيْنا هذا الاتجاه واعتبرنا رسالة تيورنج ببساطة أنها تعريف ، فإننا نثير التساؤل عما إذا كان هذا التعريف عريضا بدرجة كافية (sufficiently broad) ، بحيث أنه : هل يشمل كلَّ شئ نقوم بأدائه الآن (وقد نعمله في المستقبل) باستخدام الحواسيب ؟ الإجابة القاطعة بـ "نعم" ليست ممكنة ، ولكن الشواهد والأدلة في صالحها قوية جدا . وفيما يلي بعض الحجج **(arguments)** لقبول رسالة تيورنج على أنها تعريف العملية الحسابية الميكانيكية :

(١) أي شئ يمكن تنفيذه بحاسوب رقمي موجود يمكننا أيضا تنفيذه بواسطة آلة تيورنج .
(٢) لم يتمكن أحد للآن من عرض مسألة يمكن حلها بما نعتبره بداهة خوارزمية (an algorithm) ولا يمكننا أن نكتب لها برنامج آلة تيورنج .
(٣) تم اقتراح نماذج بديلة (alternative models) للعمليات الحسابية الميكانيكية ، ولكن لم يكن أي منها أقوى / أكثر قدرة من نموذج آلة تيورنج .

هذه الحجج لا تُعَدُّ برهانا لرسالة تيورنج ، والتي لا تزال تُعَدُّ فرضا (an assumption) . ولكن النظرة إلى رسالة تيورنج ببساطة على أنها تعريف اختياري (an arbitrary definition) تفتقد إلى نقطة هامة . حيث يمكننا اعتبار أن رسالة تيورنج تلعب في علم الحاسوب الدور نفسه الذي تلعبه القوانين الأساسية في الفيزياء والكيمياء . فالفيزياء التقليدية (classical physics) مثلا مبنية أساسا بدرجة كبيرة على قوانين نيوتن للحركة . ورغم أننا نسميها قوانين إلا أنها تفتقد إلى الضرورة المنطقية (logical necessity) ، ولكنها عبارة عن نماذج (models) مقبولة لشرح الكثير في عالمنا الفيزيائي . فنحن نقبل هذه القوانين لأن الاستنتاجات التي نصل إليها من هذه القوانين تتفق مع مشاهداتنا وخبرتنا وملاحظاتنا . ولا يمكن إثبات صحة هذه القوانين ، بل ربما يمكن بيان عدم صحتها إذا تعارضت نتيجة من نتائج التجارب

(experimental results) مع استنتاج مبنى على أساس هذه القوانين . ومن ناحية أخرى فإن الفشل المستمر في إثبات عدم صحة قانونٍ ما يقوي اعتقادنا بصحة هذا القانون وثقتنا فيه . وهذا هو الحال بالنسبة لرسالة تيورنج ، ولذلك فلدينا بعض الأسباب التي تجعلنا نعتبرها قانونا أساسيا في علم الحاسوب . فالاستنتاجات التي نصل إليها من رسالة تيورنج تتفق مع ما نعرفه عن الحواسيب الحقيقية (real computers) ، وحتى الآن فشلت جميع المحاولات لبيان عدم صحتها . وتبقى هناك دائما إمكانية أن يأتي أحدٌ في المستقبل بتعريف آخر يُفَسِّر أو يعلِّل بعض الأوضاع أو الحالات الخاصة التي لا تغطيها آلات تيورنج ولكنها تظل تقع في دائرة / مدى اصطلاحنا البديهي للعمليات الحسابية الميكانيكية . وعندئذ سيلزم تعديل بعض مناقشاتنا التالية تعديلا جوهريا . إلا أن احتمال تحقق هذه الإمكانية يبدو ضئيلا جدا .

والآن وقد قبلنا رسالة تيورنج فنحن في وضع يسمح لنا بإعطاء تعريف دقيق للخوارزمية .

تعريف ١ – ٦ :

الخوارزمية (**an algorithm**) لدالة $f : D \to R$ هي آلة تيورنج (Turing machine) M إذا أُعْطِيَت كمدخلات أي عنصر d حيث $d \in D$ على شريطها ، فإنها ستتوقف أخيراً (eventually halts) حيث تكون الإجابة الصحيحة $f(d) \in R$ على شريطها . وبصورة أكثر تحديدا يمكننا أن نتطلب أن

$$q_0 d \vdash^*_M q_f f(d), q_f \in F,$$
$$\forall\ d \in D.$$

وتعريف الخوارزمية ببرنامج آلة تيورنج يسمح لنا بأن نثبت بطريقة محكمة زَعْماً (a claim) مثل : " توجد خوارزمية ... " (there exists an algorithm …) أو " لا توجد أي خوارزمية ... " (there is no algorithm …) . إلا أن عملية إنشاء (constructing) خوارزمية صراحةً (explicitly) ولو لمسائل بسيطة نسبيا تعد عملية طويلة جدا . ولتجنب مثل هذا الأسلوب غير المريح نلجأ إلى رسالة تيورنج ونزعم أن أي شئ يمكننا تنفيذه على أي حاسوب يمكننا أيضا تنفيذه بواسطة آلة تيورنج . وبناءً عليه فيمكننا التعويض (substitution) بعبارة " برنامج بلغة C " ("C program") بدلا من " آلة تيورنج " ("Turing machine") في تعريف ١ – ٦ . وهذا سيريحنا بدرجة كبيرة من عبء كتابة وعرض خوارزميات . وعمليًّا فإننا – كما قد قمنا بذلك فعلا – سنتقدم خطوة أبعد ونقبل الأوصاف اللفظية (verbal descriptions) أو المخططات القالبية (block diagrams) كخوارزميات ، على فرض أنه يمكننا أن نكتب لها برنامج آلة تيورنج إذا طُلب منا ذلك . وهذا يؤدي إلى تبسيط النقاش

بدرجة كبيرة ، إلا أنه يتركنا بوضوح عرضة للانتقاد . فبينما يُعَدَّ الـ " برنامج بلغة C " مُعَرَّفاً ومحدَّداً تحديدا جيدا (well defined) لا يُعَدُّ كذلك " الوصف اللفظي الواضح " ("clear verbal description") ، ونصبح عندئذ عرضة لخطر الزعم بوجود خوارزميات غير موجودة . ولكن يمكننا استبعاد هذا الخطر بإمكاننا جعل النقاش بسيطا وواضحا وضوحا بديهيا ، وكذلك بإمكاننا إعطاء أوصاف موجزة ومُحْكَمَة (concise descriptions) للعمليات المعقدة نوعاً ما .

تمرينات رقم (١)

١ – ١ – أ) صمم آلة تيورنج لا تحتوي على أكثر من ثلاث حالات بحيث تقبل الآلة اللغة $L\,(a\,(a+b)^*)$. افرض أن $\Sigma = \{a, b\}$.

ب) هل يمكن تنفيذ هذا التصميم بآلة ذات حالتين (a two - state machine) ؟

١ – ٢) أنشئ آلة تيورنج تقبل اللغة التالية على $\{ a, b \}$:

أ) $L = L\,(aba^*b)$

ب) $L = \{w : |w| \text{ is even}\}$

١ – ٣) صمم آلة تيورنج توجد منتصف (middle) سلسلة (string) زوجية الطول . وبصورة أكثر تحديدا : إذا كانت السلسلة هي

$$w = a_1a_2\ldots\ a_na_{n+1}\ldots a_{2n};\quad a_i \in \Sigma$$

فإن آلة تيورنج يجب أن تعطِي / تُنتج (produces)

$$\hat{w} = a_1a_2\ldots\ a_nca_{n+1}\ldots a_{2n};\quad c \in \Gamma - \Sigma$$

١ – ٤) صمم آلة تيورنج فيها $\Gamma = \{0, 1, \square\}$ ، بحيث أنها عندما نبدأها على أي خلية (cell) تحتوي على فراغ (blank) أو 1، فإنها ستتوقف (halt) إذا وفقط إذا احتوى شريطها على 0 في موضعٍ ما عليه .

١ – ٥) لعلك لاحظت أن جميع الأمثلة في هذا الفصل اشتملت على حالة نهائية واحدة فقط . هل صحيح عموما أنه لأي آلة تيورنج توجد آلة تيورنج أخرى تحتوي على حالة نهائية واحدة فقط وتقبل اللغة نفسها (accepts the same language) ؟

١ – ٦) باستخدام أي من أجهزة جَمْع (adders) أو طرح (subtracters) أو مقارنة (comparers) أو نسخ (copiers) أو ضرب (multipliers) ارسم مخططا قالبيا (a block diagram) لآلة تيورنج تحسب الدالة التالية لأي عدد صحيح موجب ***n*** .

$$f\,(n) = n\,(n+1)$$

١ – ٧) اكتب وصفا عالي المستوى (a high - level description) لآلة تيورنج تقبل مُكَمِّل (complement) اللغة $L = \{ww^R\}$ على $\{a, b\}$. عَرِّف مجموعة من تعليمات ماكرو مناسبة تشعر أنه من السهل تنفيذها . ثم استخدمها في الحل .

١ – ٨) اكتب تنفيذاً (an implementation) للتعليمة الماكرو

$$\text{searchright } (a, q_i, q_j)$$

التي تشير إلى أن الآلة ستبحث في الشريط على يمين الموضع الحالي (current position) عن أول ظهور للرمز a . فإن وَجَدت رمزاً a قبل فراغ (a blank) فإن الآلة تذهب إلى الحالة q_i ، وما عدا ذلك فإنها تذهب إلى الحالة q_j .

١ – ٩) ذكرنا في هذا الفصل عند مناقشتنا لرسالة تيورنج أن آلات تيورنج تبدو وكأنها أقوى وأكثر قدرة من آلات الدفع لأسفل . ونظرا لأن شريط آلة تيورنج يمكننا دائما أن نجعله يتصرف مثل رَصَّة (like a stack) ، فيبدو أنه يمكننا فعليا أن نزعم أن آلة تيورنج أقوى وأكثر قدرة . ما هو العامل الهام (important factor) الذي لم نأخذه في الاعتبار في وصولنا إلى هذا الاستنتاج .

الفصل الثاني

نماذج أخرى لآلات تيورنج
Other Models of Turing Machines

تعريفنا السابق – في الفصل الأول – لآلة تيورنج القياسية ليس هو التعريف الوحيد الممكن ، بل توجد هناك تعريفات بديلة يمكن تأدية الغرض نفسه ، وبالجودة نفسها . والاستنتاجات التي يمكننا أن نصل إليها حول قدرة (power) آلة تيورنج تُعد مستقلة بدرجة كبيرة عن البنية الخاصة (specific structure) التي نختارها لها . وفي هذا الفصل نتناول بإذن الله عدة تغييرات (variations) للآلة ، ونثبت أن آلة تيورنج القياسية مكافئة – بالمفهوم الذي سنعرِّفه – للنماذج الأخرى الأكثر تعقيدا .

وإذا قَبلنا رسالة تيورنج فإننا نتوقع أن تعقيد آلة تيورنج القياسية بإعطائها وسيلة تخزين (storage device) أكثر تعقيدا سوف لا يكون له أي تأثير على قدرة الآلة . وأي حسابات (computation) يمكن إجراؤها على مثل هذه الترتيبة (arrangement) / التركيبة الجديدة ستظل تقع ضمن دائرة / طبقة (category) الحسابات الميكانيكية ، وبالتالي سيمكن تنفيذها بواسطة نموذج قياسي . ومع ذلك فإنه من المفيد لغرض توضيحي وتعليمي (instructive) أن ندرس نماذج أكثر تعقيدا ، حتى لو لم تكن لغرض آخر سوى أن العرض الصريح (explicit demonstration) للنتائج المتوقعة ستظهر لنا قوة وقدرة آلة تيورنج ، وبذلك تزداد ثقتنا في رسالة تيورنج . وهناك تغييرات عديدة يمكن إجراؤها على النموذج الأساسي المذكور في تعريف ١ – ١ . فمثلا يمكننا أن نأخذ في الاعتبار آلات تيورنج التي تحتوي على أكثر من شريط واحد ، أو على شرايط تتمدد (extend) في عدة أبعاد (several dimensions) . وسنأخذ في الاعتبار بإذن الله عدة متغيرات (variants) التي ستفيدنا في مناقشاتنا فيما بعد .

كما سنلقي نظرة بإذن الله على آلات تيورنج غير المحددة (nondeterministic) ونبيِّن أنها ليست أقوى (no more powerful) من تلك المحدَّدة . وهذا أمر غير متوقع نظرا لأن رسالة تيورنج تغطي فقط الحسابات الميكانيكية ولا تتعرض للتخمين الذكي (clever guessing) المتضمَّن (implicit) في عدم التحديد (nondeterminism) . وهناك أمر آخر لا يُحَلُّ فورا (not immediately resolved) برسالة تيورنج وهو أن تقوم آلة بتنفيذ برامج

مختلفة في أوقات مختلفة . وهذا يؤدي إلى فكرة آلة تيورنج " العامة / الشاملة " (universal) / "القابلة لإعادة البرمجة " (reprogrammable) .

وأخيرا تمهيدا للفصول التالية نلقي نظرة على الآلات المحدودة الخطية (linear bounded automata) . وهذه عبارة عن آلات تيورنج لها شريط لا نهائي (an infinite tape) ، ولكن يمكنها الاستفادة من الشريط بطريقة مقيدة فقط (only in a restricted way) .

أولا : تغييرات طفيفة على آلة تيورنج الأساسية
Minor Variations on the Basic Turing Machine

نبدأ بعون الله تعالى ببعض التغييرات الطفيفة نسبيا في تعريف ١ –١ ، ونبحث ما إذا كانت هذه التغييرات ستؤدي إلى أي اختلاف في المفهوم العام . وكلما غيَّرنا أي تعريف فإننا سنُعَرِّف نوعا جديدا من الآلات ونسأل ما إذا كانت هذه الآلات الجديدة مختلفة بمفهوم حقيقي عن تلك التي عرفناها فعليا سابقا . فماذا نقصد بالاختلاف الحقيقي / الأساسي (essential/real difference) بين طبقة آلات (class of automata) وطبقة أخرى ؟ قد تكون هناك اختلافات واضحة بين تعريفي طبقتين ، ولكن قد لا تكون لهذه الاختلافات أي تأثيرات أو نتائج مهمة لاحقة . فنعرف مثلا أن تعريفي طبقتي الآلات المحدودة المحدَّدة وغير المحدَّدة (deterministic and nondeterministic finite automata) مختلفان تماما ، ولكنَّ الطبقتين متكافئتان بمفهوم أنهما يتم التَعَرُّف عليهما بالضبط (exactly identified) بعائلة اللغات المنتظمة (family of regular languages) . وانطلاقا من هذا واستكمالاً له يمكننا تعريف التكافؤ أو عدم التكافؤ عموما بين طبقات الآلات .

تكافؤ طبقات الآلات Equivalence of Classes of Automata

حينما نُعَرِّف التكافؤ بين آلتين أو بين طبقتين من الآلات فيجب أن نحدد بدقة ماذا نقصد بهذا التكافؤ . وحتى نهاية هذا الفصل فَسَنَتَّبِع المفهوم الذي ذكرناه سابقا بالنسبة لتكافؤ الآلات dfa's والآلات nfa's ، وهو القدرة على قبول اللغات (ability to accept languages) .

تعريف ٢ – ١ :

(*) يقال إن آلتين (**two automata**) متكافئتان (**equivalent**) إذا قبلتا اللغة نفسها (same language) .

(*) ونفرض أن C_1, C_2 طبقتان من الآلات . إذا تحقق الشرط أنه لكل آلة M_1 (automaton) في الطبقة C_1 توجد آلة M_2 في الطبقة C_2 بحيث أن

$$L(M_1) = L(M_2)$$

فإننا نقول إن الطبقة C_2 هي على الأقل في قوة الطبقة C_1 (at least as powerful as) . وإذا تحقق الشرط العكسي (converse) أيضا بأنه لكل آلة M_2 في الطبقة C_2 توجد آلة M_1 في الطبقة C_1 بحيث أن

$$L(M_1) = L(M_2)$$

فإننا نقول إن الطبقتين C_1, C_2 (**the two classes**) متكافئتان (**equivalent**) .

وهناك طرق عديدة لإثبات تكافؤ الآلات ، منها مثلا الطريقة البنائية / الإنشائية (construction) لإثبات تكافؤ الآلات dfa's والآلات nfa's . ولبيان التكافؤ في حالة آلات تيورنج فإننا غالبا نستخدم الطريقة الهامة التي يُطلَق عليها " المحاكاة " (**simulation**) :

نفرض أن M آلة (an automaton) . نقول إن آلة أخرى $\hat{M}$ يمكنها محاكاة (simulating) عملية حسابية (a computation) في الآلة M إذا كانت الآلة $\hat{M}$ تستطيع أن تحاكي / تقلِّد (mimic) العملية الحسابية في M بالطريقة التالية (in the following manner) :

نفرض أن $d_0, d_1, \ldots$ هي متتابعة الأوصاف اللحظية (sequence of instantaneous descriptions) للعملية الحسابية في M ، أي أن

$$d_0 \vdash_M d_1 \vdash_M \ldots \vdash_M d_n \ldots$$

فإن الآلة $\hat{M}$ تحاكي (simulates) هذه العملية الحسابية إذا نَفَّذت

$$\hat{d}_0 \vdash^*_{\hat{M}} \hat{d}_1 \vdash^*_{\hat{M}} \ldots \vdash^*_{\hat{M}} \hat{d}_n \ldots$$

حيث $\hat{d}_0, \hat{d}_1, \ldots$ هي أوصاف لحظية ، بحيث أن كلا منها مرتبط (associated with) بتشكيلة وحيدة (a unique configuration) من تشكيلات M . وبأسلوب آخر : إذا عرفنا العملية الحسابية (computation) التي نَفَّذَتها الآلة $\hat{M}$ ، فإنه يمكننا منها أن نحدِّد بالضبط ما

هي العمليات الحسابية التي كانت ستقوم بها الآلة M إذا أُعطينا التشكيلة الابتدائية المقابلة (corresponding starting configuration) .

لاحظ أن محاكاة (simulation) حركة واحدة $d_i \vdash_M d_{i+1}$ في الآلة M قد تشتمل على عدة حركات في الآلة $\hat{M}$. والتشكيلات الوسطية في $\hat{d}_i \vdash^* \hat{d}_{i+1}$ (intermediate configurations in $\hat{M}$) قد لا تقابل (correspond to) أي تشكيلة في الآلة M ، ولكن هذا ليس له أي تأثير على أي شئ إذا استطعنا تحديد التشكيلات ذوات العلاقة في الآلة $\hat{M}$ (relevant configurations of) . وطالما أننا نستطيع من حسابات الآلة $\hat{M}$ تحديد ماذا كانت ستفعل الآلة M ، فإن المحاكاة صحيحة . وإذا استطاعت الآلة $\hat{M}$ محاكاة أي عملية حسابية في الآلة M ، فإننا نقول إن الآلة $\hat{M}$ يمكنها محاكاة الآلة M . ويجب أن يكون واضحا أنه إذا استطاعت الآلة $\hat{M}$ محاكاة الآلة M ، فإنه يمكننا عمل الترتيبات اللازمة حتى تقبل الآلتان $M, \hat{M}$ اللغة نفسها ، وتصبح الآلتان متكافئتين (equivalent) . ولبيان تكافؤ طبقتين من الآلات ، علينا أن نثبت أنه لكل آلة في إحدى الطبقتين توجد آلة في الطبقة الأخرى قادرة على محاكاتها .

آلات تيورنج ذوات اختيار البقاء (في الموضع)
Turing Machines with a Stay - Option

عند تعريف آلة تيورنج القياسية ذكرنا أن رأس القراءة والكتابة يجب أن تتحرك إما لليمين أو لليسار . أحيانا يكون من المناسب أن يكون هناك اختيار ثالث وهو أن تبقى رأس القراءة والكتابة في موضعها بعد إعادة كتابة (rewriting) محتويات الخلية . وهكذا يمكننا تعريف آلة تيورنج ذات اختيار البقاء (stay - option) بأن نستبدل بدَالَّةِ الانتقال δ في تعريف ١ – ١ الدالة

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$$

حيث S تعني عدم حركة رأس القراءة والكتابة . وهذا الاختيار لا يزيد من قدرة الآلة .

نظرية ٢ – ١ :

طبقة آلات تيورنج ذوات اختيار البقاء مكافئة لطبقة آلات تيورنج القياسية .

البرهان :

(i) **نظرا لأن آلة تيورنج ذات اختيار البقاء هي امتداد** (extension) **للنموذج القياسي** (standard model) **، فمن الواضح أن أي آلة تيورنج قياسية سيمكن محاكاتها بآلة ذات اختيار البقاء .**

(ii) **ولبيان العكس : نفرض أن** $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ **هي آلة تيورنج ذات اختيار البقاء المراد محاكاتها بآلة تيورنج قياسية** $\hat{M} = \left(\hat{Q}, , \Sigma, \Gamma, \hat{\delta}, \hat{q_0}, \square, \hat{F}\right)$ **. لكل حركة في الآلة** $\boldsymbol{M}$ **تقوم الآلة المحاكية** $\hat{\boldsymbol{M}}$ (simulating machine) **بما يلي . إذا لم تشتمل حركة** $\boldsymbol{M}$ **على اختيار البقاء ، فإن الآلة المحاكية تعمل حركة واحدة مطابقة** (identical) **أساسا للحركة المراد محاكاتها . أما إذا اشتملت حركة** $\boldsymbol{M}$ **على** $\boldsymbol{S}$ **، فإن** $\hat{\boldsymbol{M}}$ **ستعمل حركتين : الأولى تعيد كتابة الرمز** (rewrites the symbol) **وتُحَرِّك رأس القراءة والكتابة لليمين ، والثانية تُحَرِّك رأس القراءة والكتابة لليسار ، تاركةً محتويات الشريط دون تغيير . ويمكننا إنشاء** (constructing) **الآلة المحاكية من الآلة** $\boldsymbol{M}$ **بتعريف** $\hat{\delta}$ **كما يلي :**

لأي انتقال (transition)

$$\delta(q_i, a) = (q_j, b, L \text{ or } R)$$

نضع في $\hat{\delta}$

$$\hat{\delta}\left(\hat{q}_i, a\right) = \left(\hat{q}_i, b, L \text{ or } R\right)$$

ولأي انتقال - S (S - transition)

$$\delta(q_i, a) = (q_j, b, S)$$

نضع في $\hat{\delta}$ **الانتقالات المقابلة**

$$\hat{\delta}\left(\hat{q}_i, a\right) = \left(\hat{q}_{js}, b, R\right)$$

و

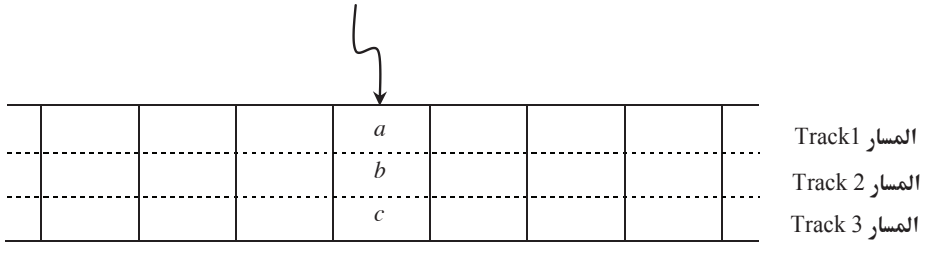
$$\hat{\delta}\left(\hat{q}_{js}, c\right) = \left(\hat{q}_j, c, L\right)$$

وذلك $\forall c \in \Gamma$.

ومن الواضح أن أي عملية حسابية في M سيكون لها عملية حسابية مقابلة في $\hat{M}$ ، وهكذا فإن $\hat{M}$ يمكنها محاكاة M .

والمحاكاة تُعَدُّ طريقة قياسية لإثبات تكافؤ الآلات ، والصيغة التي وصفناها آنفا تجعل من الممكن – كما رأينا في النظرية السابقة – الحديث عن عملية التكافؤ وإثبات نظريات عن التكافؤ . وفي دراستنا التالية سنستخدم بإذن الله مرارا اصطلاح المحاكاة ، ولكننا عموما سوف لا نحاول وصف كل شئ بطريقة مفصلة ومحكمة . فبالنسبة لآلات تيورنج تُعد المحاكاة التامة أمرا مُمِلا ومرهقا. ولتجنب ذلك ستقتصر مناقشاتنا على الطريقة الوصفية (descriptive) بدلا من طريقة برهنة النظريات . وسنعطي بإذن الله الخطوط العريضة للمحاكاة ، ولكنه ليس من الصعب أن نرى كيفية تحويل هذه الخطوط العريضة إلى خطوات محكمة (rigorous) . وسيجد القارئ الكريم أنه من المفيد من الناحية التعليمية (instructive) تحويل كل محاكاة إلى مخطط (sketch) بلغةٍ عالية المستوى (a higher - level language) أو بشبه كود (pseudocode) .

وقبل الانتقال إلى تعريف نماذج أخرى لآلات تيورنج نود إبداء ملاحظة على آلة تيورنج القياسية . تعريف ١ – ١ يشمل ضمنيا (implicitly) أن يكون أي رمز شريط (tape symbol) مركبا من عدة رموز (a composite of characters) بدلا من رمز واحد فقط (a single one) . ويمكن توضيح ذلك صراحةً (explicitly) برسم صيغة ممتدة / موسَّعة (extended version) لشكل ١ – ١ ، بحيث تكون رموز الشريط ثلاثيات (triplets) من رموز أبجدية أبسط (simpler alphabet) ، كما يوضح ذلك شكل ٢ – ١ .



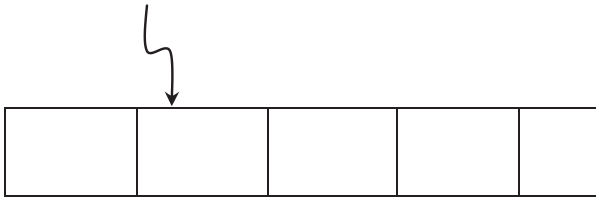
شكل ٢–١

وفي هذا الشكل قَسَّمنا كل خلية من خلايا الشريط إلى ثلاثة أجزاء تُدْعَى " مسارات " (tracks) ، يحتوي كل منها على عنصر واحد من الثلاثية (triplet) . وبناءً على هذه الصيغة / النظرة ، فمثل هذه الآلة يطلق عليها أحيانا " آلة تيورنج ذات المسارات المتعددة " (multiple tracks) . ولكن هذه النظرة لا توسِّع / لا تمدِّد بأي حال تعريف ١ – ١ ، نظرا لأن كل ما نحتاج عمله هو أن نجعل Γ مجموعة أبجدية (an alphabet) يتكون فيها كل رمز من أجزاء متعددة .

ولكن هناك نماذج أخرى من آلات تيورنج تتضمن تغييرا في التعريف ، بحيث أنه يجب علينا بيان التكافؤ مع الآلة القياسية . وفيما يلي نلقي نظرة على نموذجين من هذه النماذج ، واللذين يُستخدمان أحيانا على أنهما التعريف القياسي (the standard definition) . وفي التمرينات بنهاية الفصل سنرى بإذن الله نماذج أخرى أقل شيوعا .

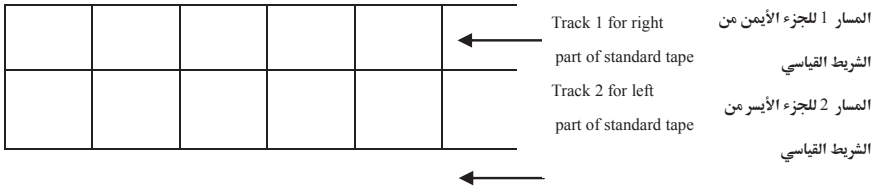
آلات تيورنج ذوات الشريط النصف لانهائي
Turing Machines with Semi – Infinite Tape

كثير من المؤلفين لا يعتبرون النموذج المبين في شكل ١ – ١ قياسيا ، وإنما يستخدمون نموذجا ذا شريط غير محدود (unbounded) في اتجاه واحد فقط . ويمكننا أن نصور ذلك كشريط له حد أيسر (a left boundary) [انظر شكل ٢ – ٢] . وآلة تيورنج هذه – فيما عدا ذلك – تعد مطابقة لنموذجنا القياسي ، غير أنه لا يُسمح بتحرك لليسار (left move) عندما تكون رأس القراءة والكتابة عند الحدود (at the boundary) .



شكل ٢–٢

وليس من الصعب أن نرى أن هذا القيد لا يؤثر على قدرة الآلة . ولمحاكاة آلة تيورنج قياسية M بواسطة آلة $\hat{M}$ ذات شريط نصف لا نهائي نستخدم النموذج المبين في شكل ٢ – ٣ .



شكل ٢-٣

الآلة المحاكية $\hat{M}$ لها شريط ذو مسارين . المسار العلوي يحتوي على المعلومات الموجودة يمين نقطة إسنادٍ ما (some reference point) على شريط M . ونقطة الإسناد / النقطة المرجع (reference point) قد تكون مثلا موضع رأس القراءة والكتابة عند بداية العملية الحسابية (start of the computation) . وأما المسار السفلي فيحتوي على الجزء الأيسر من شريط M بترتيب عكسي . والآلة $\hat{M}$ تُبَرْمَج بحيث أنها تَستخدم المعلومات الموجودة على المسار العلوي فقط طالما أن رأس القراءة والكتابة في الآلة M موجودة يمين نقطة الإسناد ، وتعمل على المسار السفلي طالما أن M تتحرك في الجزء الأيسر من شريطها . ويمكن عمل هذا التميُّز (distinction) بتجزئة (partitioning) مجموعة حالات(state set of) $\hat{M}$ إلى جزئين :

Q_U, Q_L ، حيث نستخدم الجزء Q_U حين العمل على المسار العلوي ، ونستخدم Q_L على المسار السفلي . وتوضع علامات خاصة للنهاية # (special end markers) على الحدود اليسرى (left boundary) للشريط لتسهيل التحوُّل (switching) من مسار لآخر .

مثال ٢ – ١ :

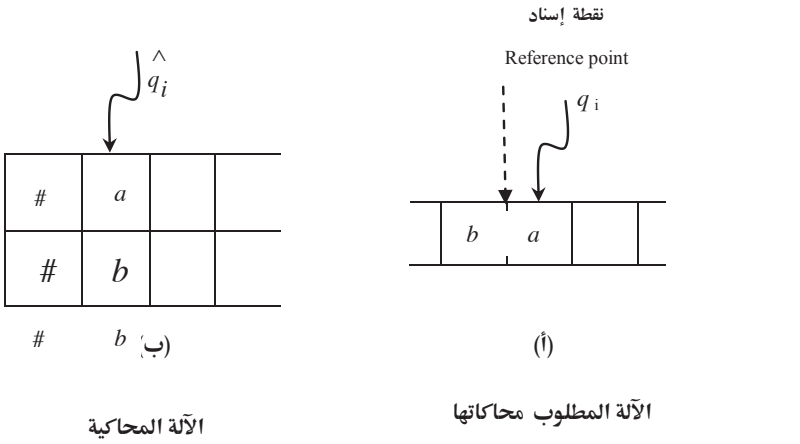
نفرض أن الآلة المطلوب محاكاتها والآلة المحاكية هما المبينتان في تشكيلتي (configurations) شكل ٢ – ٤ .

ونفرض أن الحركة المطلوب محاكاتها يولِّدها (generated by) الانتقال

$$\delta(q_i, a) = (q_j, c, L)$$

الآلة المحاكية ستقوم أولا بالحركة عبر الانتقال

$$\hat{\delta}\left(\hat{q}_i,(a,b)\right)=\left(\hat{q}_j,(c,b),L\right)$$

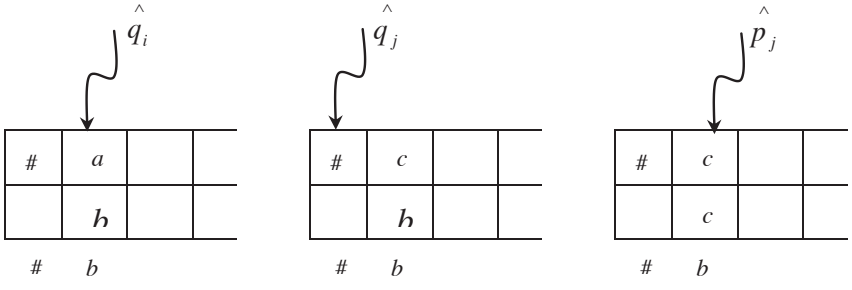


شكل ٢-٤

حيث $\hat{q}_i \in Q_U$. ونظرا لأن $\hat{q}_i$ تنتمي إلى Q_U فإن المعلومات الموجودة في المسار العلوي فقط هي التي تؤخذ في الاعتبار في الوقت الحالي . والآن فإن الآلة المحاكية ترى (#, #) في الحالة $\hat{q}_j \in Q_U$. وبعد ذلك تَستخدم انتقالا

$$\hat{\delta}\left(\hat{q}_j, (\#, \#)\right) = \left(\hat{p}_j, (\#, \#), R\right)$$

حيث $\hat{p}_j \in Q_L$ ، فتضع الآلة في التشكيلة المبينة في شكل ٢ – ٥ .



شكل ٢ – ٥

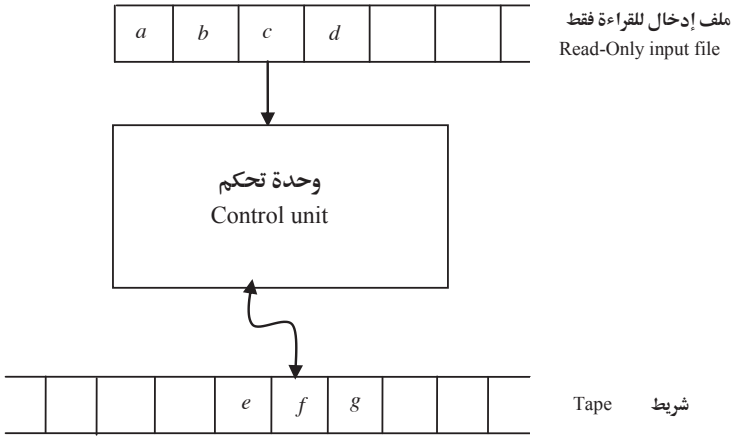
متتابعة التشكيلات عند محاكاة $\delta(q_i, a) = (q_j, c, L)$

والآن تكون الآلة في حالة من حالات Q_L ، وتعمل على المسار السفلي . والتفاصيل التي تلي ذلك في عملية المحاكاة مباشرة ويسهل تتبعها .

آلة تيورنج المفصولة عن الخط The Off - Line Turing Machine

التعريف العام للآلة ذاتية الحركة (automaton) [انظر شكل ١ – ١] يشتمل على ملف إدخال (input file) وتخزين مؤقت (temporary storage) . وفي تعريف ١ – ١ لآلة تيورنج أهملنا ملف الإدخال من أجل تبسيط الأمور ، مُدَّعين أن ذلك لا يؤثر على مفهوم آلة تيورنج . والآن نناقش هذا الادعاء .

إذا أَعَدْنا ملف الإدخال إلى تمثيل الآلة ، فإننا نحصل على ما يُسمَّى " آلة تيورنج المفصولة عن الخط " (**off - line Turing machine**) . وفي هذه الآلة فإن كل حركة (move) تحكمها (governed by) : الحالة الداخلية (internal state) ، وما يُقرأ حاليا (currently read) من ملف الإدخال ، وما تراه رأس القراءة والكتابة . ويوضح شكل ٢ – ٦ مخططا لتمثيل (schematic representation) آلة مفصولة عن الخط .

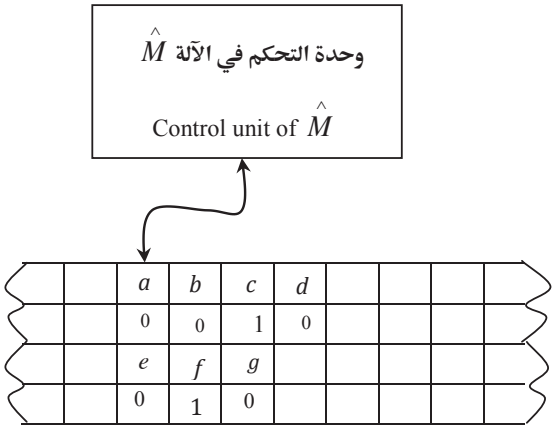


شكل ٢ – ٦

ويمكن بسهولة إعطاء التعريف الشكلي (formal definition) لآلة تيورنج المفصولة عن الخط ، ولكننا سنتركه كتدريب للقارئ الكريم . وما نود أن نقوم به باختصار هو الإشارة إلى سبب التكافؤ بين طبقة آلات تيورنج المفصولة عن الخط وطبقة الآلات القياسية .

نلاحظ أولا أنه يمكننا محاكاة سلوك (behavior) أي آلة تيورنج قياسية بنموذج مفصول عن الخط (some off – line model) . كل ما تحتاج أن تعمله الآلة المحاكية هو أن تنسخ المدخلات من ملف الإدخال إلى الشريط . ومن ثَمَّ يمكنها التقدم والاستمرار في العمل بالطريقة نفسها كالآلة القياسية .

وأما محاكاة آلة مفصولة عن الخط M (off – line machine) بآلة قياسية $\hat{M}$ فإن ذلك يتطلب وصفا أطول . ويمكن لآلة قياسية أن تحاكي العملية الحسابية لآلة مفصولة عن الخط باستخدام التشكيلة / الترتيبة المكونة من أربعة مسارات (four – track arrangement) والمبينة في شكل ٢ – ٧ . ومحتويات الشريط التي تظهر في هذا الشكل تمثل التشكيلة المعينة



شكل ٢– ٧

(specific configuration) التي يبينها مخطط شكل ٢ – ٦ ولكل مسار من المسارات الأربعة في الآلة $\hat{M}$ دور محدَّد في المحاكاة . فالمسار الأول يحتوي على المدخلات (input) ، والمسار الثاني يحدد الموضع الذي نقرأ عنده المدخلات ، والمسار الثالث يمثل شريط الآلة M ، والمسار الرابع يبين موضع رأس القراءة والكتابة في الآلة M .

ومحاكاة أي حركة في الآلة M يتطلب عددا من الحركات في الآلة $\hat{M}$. فإذا بدأنا من موضع قياسيٍّ ما – وليكن النهاية اليسرى – وباستخدام المعلومات ذات الصلة (relevant information) التي تشير إليها (marked by) علامات خاصة للنهاية ، فإن الآلة $\hat{M}$ تبحث في المسار 2 لتحديد الموضع الذي نقرأ عنده ملف الإدخال في الآلة M . ويتم تَذكُّر (remembering) الرمز الذي وجدناه في الخلية المقابلة (corresponding cell) في المسار 1 بوضع وحدة التحكم في الآلة $\hat{M}$ في حالة (state) نختارها لهذا الغرض . وبعد ذلك يتم البحث في المسار 4 عن موضع رأس القراءة والكتابة في الآلة M . والآن بمعلومية المُدْخَل الذي نتذكره (remembered input) والرمز الموجود في المسار 3 نعلم أن الآلة M عليها أن تعمل . ومرة أخرى هذه المعلومات تتذكرها الآلة $\hat{M}$ بحالة داخلية (internal state) مناسبة . وبعد ذلك يتم تعديل (modifying) جميع المسارات الأربعة في شريط الآلة $\hat{M}$ لتعكس حركة الآلة M . وأخيرا تعود رأس القراءة والكتابة في الآلة $\hat{M}$ إلى الموضع القياسي لمحاكاة الحركة التالية .

ثانيا : آلات تيورنج ذوات تخزين أكثر تعقيدا
Turing Machines with more Complex Storage

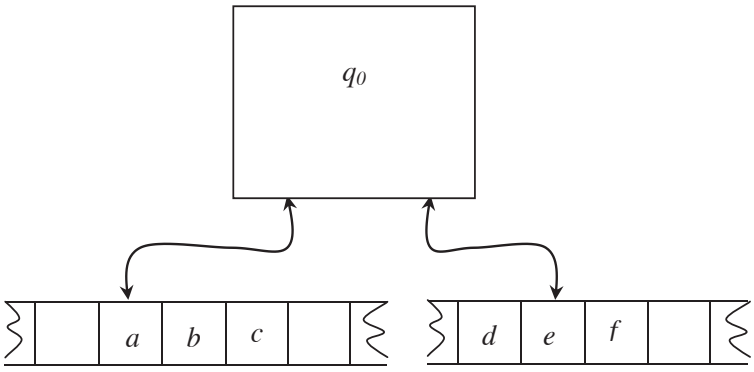
أداة التخزين في آلة تيورنج القياسية بسيطة جدا لدرجة أن المرء قد يفكر أنه من الممكن زيادة قدرة الآلة باستخدام معدات تخزين أكثر تعقيدا . ولكن الواقع أن هذا غير صحيح ، كما سنوضح ذلك بأذن الله بمثالين .

آلات تيورنج متعددة الأشرطة Multitape Turing Machines

آلة تيورنج متعددة الأشرطة هي آلة تيورنج لها عدة أشرطة ، لكل شريط منها رأس قراءة وكتابة خاصة به ، ويتم التحكم فيها بطريقة مستقلة (independently controlled) عن الرؤوس الأخرى [انظر شكل ٢ – ٨] .

والتعريف الشكلي لآلة تيورنج متعددة الأشرطة يزيد على تعريف ١ – ١ ، نظرا لأنه يتطلب دالة انتقال معدَّلة . ونمطيا (typically) نُعَرِّف آلة ذات n شريط (n-tape machine) هكذا :

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$$



شكل ٢– ٨

حيث $Q, \Sigma, \Gamma, q_0, F$ كما هي في تعريف ١ – ١ ، ولكن دالة الانتقال

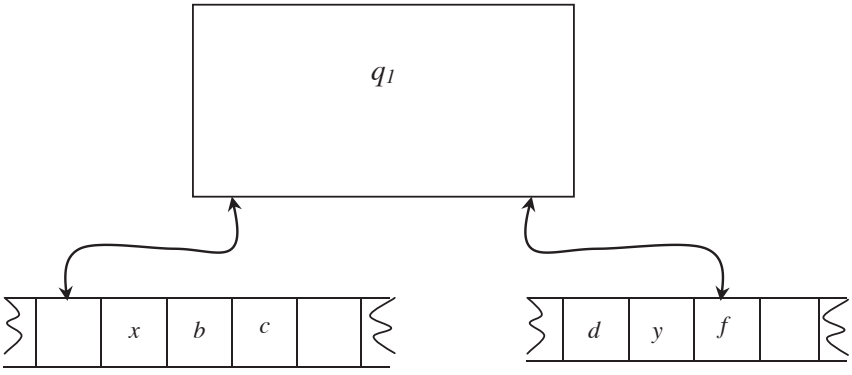
$$\delta : Q \times \Gamma^{n} \to Q \times \Gamma^{n} \times \{L, R\}^{n}$$

تحدِّد ماذا يحدث على جميع الأشرطة . فمثلا إذا كانت $n = 2$ وكانت التشكيلة الحالية (current configuration) هي تلك المبيَّنة في شكل ٢ – ٨ ، فإن

$$\delta(q_0, a, e) = (q_1, x, y, L, R)$$

نُفسِّرها كما يلي : قاعدة الانتقال (transition rule) يمكن تطبيقها فقط إذا كانت الآلة في الحالة q_0 ، وكانت رأس القراءة والكتابة الأولى ترى رمزا a ، والثانية ترى رمزا e . ثم تستبدل بالرمز الموجود على الشريط الأول رمزا x ، وتتحرك رأس القراءة والكتابة الخاصة به إلى اليسار. وفي الوقت نفسه فإن الرمز الموجود على الشريط الثاني تعاد كتابته (rewritten) كرمز y ، وتتحرك رأس القراءة والكتابة إلى اليمين . ومن ثَمَّ تُغيِّر وحدة التحكم حالتها إلى q_1 ، وتنتقل الآلة إلى التشكيلة الجديدة المبينة في شكل ٢ – ٩ .

ولبيان التكافؤ بين آلات تيورنج متعددة الأشرطة وآلات تيورنج القياسية فإننا نزعم أن أي آلة تيورنج متعددة الأشرطة M يمكن محاكاتها بآلة تيورنج قياسية $\hat{M}$ ، وبالعكس فإن أي آلة



شكل ٢– ٩

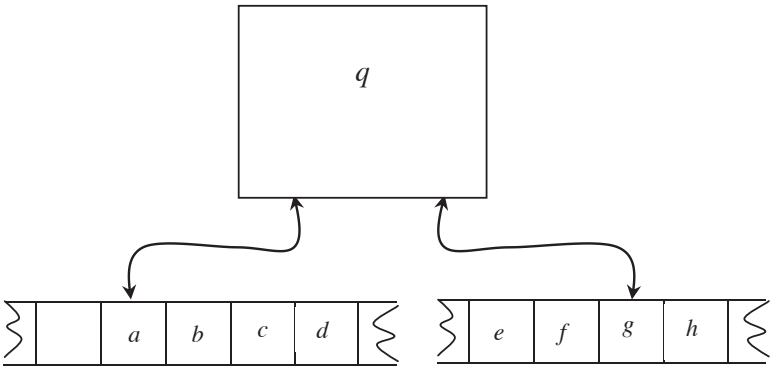
تيورنج قياسية يمكن محاكاتها بآلة تيورنج متعددة الأشرطة . الجزء الأخير من هذا الزعم لا يحتاج لأي تفصيل ، نظرا لأنه يمكننا دائما اختيار تشغيل آلة متعددة الأشرطة بشريط واحد فقط من أشرطتها بحيث يقوم بالعمل المفيد . وأما محاكاة آلة متعددة الأشرطة بآلة ذات شريط واحد فتُعد أعقد قليلا ولكنها مباشرة وسهلة الفهم .

مثال ٢ – ٢ :

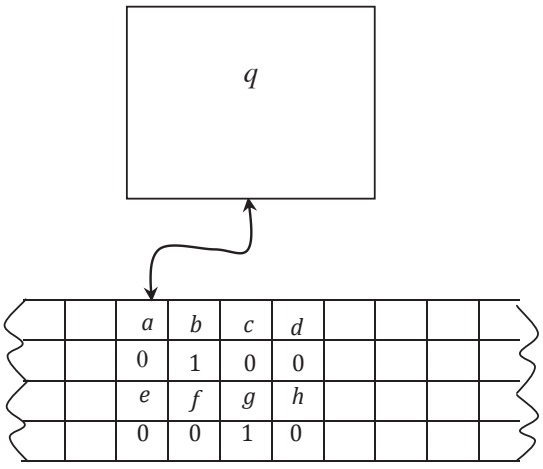
نفرض أن لدينا الآلة ذات الشريطين (two-tape machine) المبينة في شكل ٢ – ١٠ . الآلة المحاكية ذات الشريط الواحد ستحتوي على أربعة مسارات [انظر شكل ٢ – ١١] . المسار الأول يمثل محتويات الشريط 1 في الآلة M . الجزء غير الفارغ (nonblank part) في المسار الثاني يحتوي كله على أصفار (zeros) باستثناء 1 وحيد يدل على موضع رأس القراءة والكتابة في الآلة M . المساران 3, 4 يلعبان دورا شبيها بالنسبة للشريط 2 في الآلة M . وشكل ٢ – ١١ يجعل من الواضح أنه بالنسبة لتشكيلات $\hat{M}$ التي لها هذه الصيغة المبينة توجد تشكيلة مقابلة وحيدة للآلة M .

وتمثيل آلة متعددة الشرائط بآلة ذات شريط واحد يشبه التمثيل المستخدم في محاكاة آلة مفصولة عن الخط . والخطوات الفعلية في عملية المحاكاة هي تقريبا نفسها ، والاختلاف الوحيد هو أننا نأخذ في الاعتبار أشرطة أكثر . والمخطط (outline) الذي ذكرناه سابقا لمحاكاة الآلات

المفصولة عن الخط يمكن تطبيقه في حالتنا هنا مع بعض التعديلات البسيطة ، ويشير هذا المخطط



شكل ٢– ١٠



شكل ٢– ١١

إلى إمكانية كتابة إجراء (procedure) لإنشاء (constructing) دالة انتقال $\hat{\delta}$ للآلة $\hat{M}$ من دالة الانتقال δ في الآلة M. وليس من الصعب جعل عملية الإنشاء هذه دقيقة ومحكمة (precise) ، إلا أنها ستكون طويلة في كتابتها . وبالتأكيد فإن العمليات الحسابية في الآلة $\hat{M}$

ستظهر طويلة وتفصيلية ، ولكن هذا لا يؤثر على الاستنتاج : أي شئ يمكن تنفيذه على الآلة M يمكن أيضا تنفيذه على الآلة $\hat{M}$.

وهناك نقطة هامة يجب أن تكون واضحة في أذهاننا ، وهي أنه عندما نقول إن آلة تيورنج ذات الأشرطة المتعددة ليست أقوى (ليست أقدر) (no more powerful) من آلة تيورنج القياسية فإننا نقصد بهذه العبارة فقط ما يمكن تنفيذه بهاتين الآلتين ، وخاصة : ما هي اللغات التي يمكن أن تُقْبَل .

مثال ٢ – ٣ :

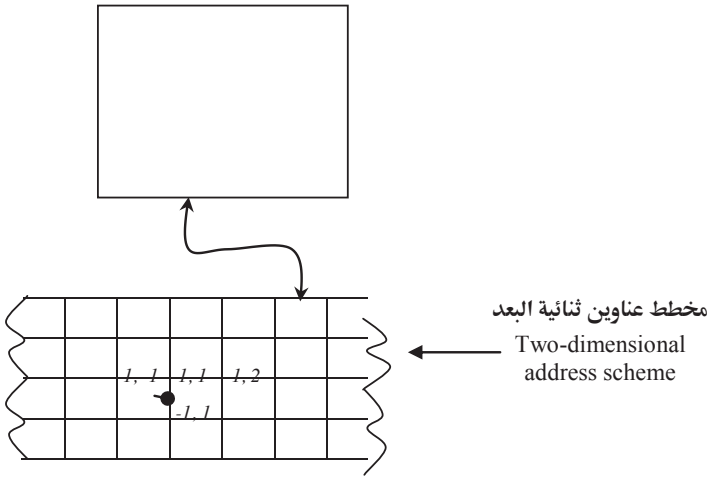
نفرض أن لدينا اللغة $\{a^n b^n\}$. في مثال ١ – ٨ وَصَفْنا طريقة طويلة لقبول هذه اللغة بآلة تيورنج ذات شريط واحد . وباستخدام آلة ذات شريطين تكون المهمة أسهل بكثير . افرض أن سلسلة ابتدائية $a^n b^m$ (initial string) قد كُتبت على الشريط 1 عند بداية العملية الحسابية . نقرأ حينئذ جميع الرموز $a's$ ناسخين (copying) إياها على الشريط 2 . وعندما نصل إلى نهاية الرموز $a's$ فإننا نوائم (match) بين الرموز $b's$ الموجودة على الشريط 1 والرموز $a's$ المنسوخة (copied) على الشريط 2 . وبهذه الطريقة يمكننا أن نحدد ما إذا كان هناك عدد مساو من الرموز $a's$ والرموز $b's$ دون الحاجة إلى تكرار حركة رأس القراءة والكتابة للأمام والخلف (repeated back – and – forth movement) .

* * *

تذكَّر أن النماذج المتعددة من آلات تيورنج تُعتبر متكافئة فقط بالنسبة لقدرتها على تنفيذ أشياء معينة ، وليس بالنسبة لسهولة البرمجة أو أي مقاييس كفاءة أخرى قد نأخذها في الاعتبار . وسنعود بإذن الله تعالى إلى هذه النقطة في الفصل الأخير (السادس) .

آلات تيورنج متعددة الأبعاد Multidimensional Turing Machines

آلة تيورنج متعددة الأبعاد هي آلة يمكن النظر فيها إلى الشريط على أنه يمتد إلى اللانهاية في أكثر من بُعد (dimension) . ويبيِّن شكل ٢ – ١٢ مخططا لآلة تيورنج ثنائية البعد (two – dimensional) .



شكل ٢– ١٢

والتعريف الشكلي لآلة تيورنج ثنائية البعد يشتمل على دالة انتقال δ صيغتها :

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, U, D\}$$

حيث U تحدد حركة رأس القراءة والكتابة لأعلى ، بينما D تحدد حركتها لأسفل .

ولمحاكاة هذه الآلة على آلة تيورنج قياسية يمكننا استخدام النموذج ثنائي المسار (two-track model) المبيَّن في شكل ٢ – ١٣، حيث نقوم أولا بإعطاء / بإلحاق (associating)

	a				*b*						
	1	#	2	#	1	0	#	-	3	#	

شكل ٢– ١٣

ترتيب (an ordering) أو عنوان (address) لكل من خلايا (cells) الشريط ثنائي البعد (two-dimensional tape) . ويمكن تنفيذ ذلك بعدة طرق كالمبينة مثلا في الصيغة ثنائية البعد في شكل ٢ – ١٢ ، حيث يقوم الشريط ثنائي المسار في الآلة المحاكية باستخدام

أحـد المسارين لتخـزين محتويات الخـلايا والآخر للاحتفاظ بالعنوان المرافق . وفـي مخـطط شكل ٢ – ١٢ : التشكيلة (configuration) التي فيها الخلية (1,2) تحتوي على a ، والخلية (10,-3) تحتـــوي علـــى b تظهـــر في شـــكل ٢ – ١٣ . ولاحـــظ أن أحـــد التعقيـــدات (complications) التي نواجهها هي أن عنـوان الخلية يمكن أن يحتوي على أعداد صحيحة كـبيرة كـبرا اختياريـا (arbitrarily large integers) ، وبالتـالي لا يستطيـع مـسار العنـوان (address track) استخدام مجال ثابت السعة (fixed – size field) لتخزين العناوين . وبدلا من ذلك علينا أن نستخدم ترتيبة لمجال متغير السعة ، مستخدمين بعض الرموز الخاصة لبيان حدود المجالات (delimit the fields) كما هو مبين في الشكل .

نفـرض أنـه عند بدايـة محاكـاة أي حـركة تكـون رأس القـراءة والكتابـة في الآلة ثنائية البعد M ، ورأس القـراءة والكتابة في الآلـــة المحاكيـــة $\hat{M}$ دائمـا عنـد خليـتين متقـابلتين (on corresponding cells) . لمحاكاة أي حركة فإن الآلة المحاكية $\hat{M}$ تحسب أولا عنوان الخلية التي ستتحرك إليها الآلة M . وهذه تكون عملية حسابية بسيطة باستخدام مخطط العناوين ثنائية البعد . وبمجرد حساب العنوان فإن الآلة $\hat{M}$ تبحث عن الخلية التي هذا عنوانها على المسار 2 ، ثم تُغَيِّر محتويات الخلية بما يوائم حركة الآلة M . ومرة أخرى إذا أُعطينا الآلة M فهناك طريقة مباشرة لبناء (construction) الآلة $\hat{M}$.

ثالثا : آلات تيورنج غير المحدَّدة Nondeterministic Turing Machines

بينما تجعلنا رسالة تيورنج نقبل أن بِنية الشريط الخاصة (specific tape structure) لا تـؤثر علـى قـدرة آلـة تيـورنج ، إلا أننـا لا نسـتطيع أن نقـول الشـئ نفسـه بالنسـبة لعـدم التحديـد (nondeterminism) . وذلك لأن عدم التحديد يشتمل على عنصر الاختيار وبالتالي له صبغة / طبيعة / نكهة غير آلية / ميكانيكية (nonmechenistic flavor) ، وبناءً عليه فإن اللجوء إلى رسالة تيورنج غير مناسـب . ويجـب علينا أن ننظـر بتفصيـل أكثر إلى تأثير عـدم التحديد إذا أردنا أن نزعـم أن عـدم التحديـد لا يضيـف شيئا إلى قـدرة آلـة تيـورنج . ومـرة أخـرى نلجـأ إلى المحاكـاة ، حيث نثبت أن سلوك عدم التحديد يمكن معالجته بصورة تحديدية (handled deterministically) .

تعريف ٢ –٢ :

آلة تيورنج غير المحدَّدة (nondeterministic Turing machine) **هي آلة ذاتية الحركة كتلك المعطاة في تعريف ١ – ١ ، غير أن دالة الانتقال δ هي الآن دالة**

$$\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L, R\}}$$

وكما هو الحال دائما عندما يكون لدينا عدم تحديد ، فإن مدى δ (range of) **يكون عبارة عن مجموعة من الانتقالات المحتملة يمكن للآلة أن تختار أياً منها .**

مثال ٢ – ٤ :

إذا كان لآلة تيورنج انتقالات تحددها العلاقة

$$\delta (q_0, a) = \{(q_1, b, R) , (q_2, c, L)\}$$

فإن هذه الآلة غير محددة (nondeterministic) .

والحركتان

$$q_0aaa \vdash bq_1aa$$

و

$$q_0aaa \vdash q_2 \square caa$$

ممكنتان .

* * *

ونظرا لأنه ليس من الواضح ما هو الدور الذي يلعبه عدم التحديد في حساب الدوال ، فإن الآلات غير المحددة يُنظر إليها عادة على أنها آلات قبول (accepters) **. ويُقال لآلة تيورنج غير محددة إنها تقبل سلسلة w إذا كانت هناك أي متتابعة تحركات محتملة** (possible sequence of moves) **بحيث أن**

$$q_0w \vdash^* x_1\, q_f\, x_2$$

حيث $q_f \in F$. والآلة غير المحددة قد تحتوي على تحركات ممكنة تؤدي إلى حالة غير نهائية أو إلى عروة لا نهائية . ولكن – كما هو الحال دائما في حالة عدم التحديد – هذه البدائل (alternatives) لا تهمنا ، وكل ما نهتم به هو وجود متتابعة ما من التحركات تؤدي إلى القبول (acceptance) .

ولإثبات أن آلة تيورنج غير المحددة ليست أقدر / أقوى من آلة تيورنج المحددة ، نحتاج لإعطاء مكافئ محدَّد لعدم التحديد (a deterministic equivalent for the nondeterminism) . وقد ألمحنا (alluded to) فعلاً من قبل إلى حالة كهذه . حيث يمكن النظر إلى عدم التحديد على أنه خوارزمية محدَّدة لتعقُّب المسار العكسي (أي من الخلف للأمام أي بالرجوع من حيث أتينا) (a deterministic backtracking algorithm) ، ويمكن لآلة محددة أن تحاكي آلة غير محددة طالما أنها تستطيع معالجة مسألة الاحتفاظ بالسجلات (handle the bookkeeping) المطلوبة في عملية تعقب المسار العكسي (من الخلف للأمام) (backtracking) . وكي ترى كيف يمكن تنفيذ ذلك ببساطة ، دعنا ننظر إلى عدم التحديد نظرة بديلة تفيدنا في كثير من المناقشات والدراسات . حيث يمكننا النظر إلى أي آلة غير محددة على أنها آلة لديها القدرة على تكرير / نسخ (replicating) نفسها كلما كان ذلك ضروريا . وعندما يكون هناك أكثر من حركة واحدة ممكنة ، فإن الآلة تنتج عددا قدر الحاجة من هذه الآلات المكررة (النسخ المطابقة) (replicas) ، وتعطي كل واحدة منها مهمة تنفيذ واحدة من هذه الحركات البديلة . وهذه النظرة لعدم التحديد قد تبدو بصورة خاصة غير آلية / ميكانيكية (nonmechanistic) ، نظرا لأن النسخ / التكرير غير المحدود (unlimited replication) هو بالتأكيد ليس ضمن قدرات الحواسيب في الوقت الحاضر . ومع ذلك فإن المحاكاة ممكنة .

وإحدى طرق تصوير هذه المحاكاة هي استخدام آلة تيورنج قياسية تحتفظ بجميع الأوصاف اللحظية الممكنة (possible instantaneous descriptions) للآلة غير المحدَّدة على شريطها ، مفصولة عن بعضها البعض بعلامةٍ / باصطلاحٍ ما (separated by some convention) . والشكل التالي (شكل ٢ – ١٤) يوضح طريقة يمكن أن يظهر فيها التشكيلان aq_0aa و bbq_1a . وفي هذا الشكل نستخدم الرمز × ليحد (delimit) منطقة اهتمامنا (area of interest) ، بينما نستخدم الرمز + ليفصل بين الأوصاف اللحظية الفردية (individual) .

	□	×	a	q_0	a	a	+	b	b	q_1	a	×	□	

شكل ٢ـ ١٤

وتقـوم الآلـة المحاكيـة بـالنظر إلى جميـع التشـكيلات الفعالـة / النشـطة (active configurations) وتحديثها (updating) بناءً على برنامج الآلة غير المحددة . وسوف تشتمل التشـكيلات الجديـدة أو توسـيع (expanding) الأوصـاف اللحظيـة علـى تحريـك (moving) الرمـوز / العلامـات (markers) × . وبالتأكيـد سـتكون التفاصيل مملـة وطويلـة ، ولكـن ليس مـن الصعب تصورها . وعلى أساس هذه المحاكاة فإننا نستنتج أنه لأي آلة تيورنج غير محددة توجد آلة قياسية محددة مكافئة .

نظرية ٢ – ٢ :

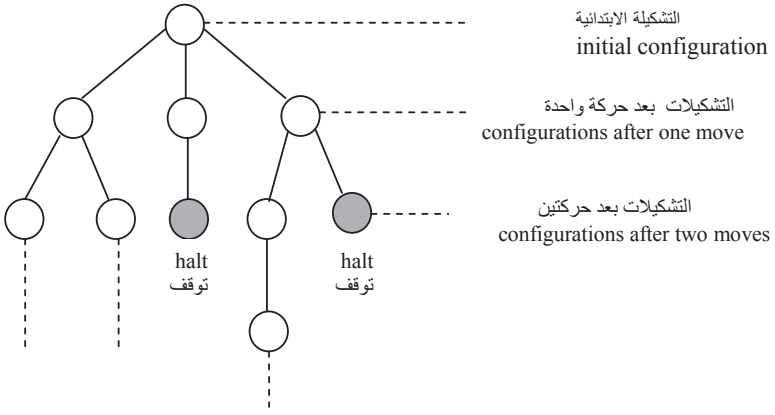
طبقة آلات تيورنج المحددة وطبقة آلات تيورنج غير المحددة متكافئتان .

البرهان :

استخدم طريقـة البنـاء (construction) المـذكورة قبـل النظريـة لإثبـات أن أي آلـة تيورنج غير محددة يمكن محاكاتها بآلة محددة .

* * *

سنعود بإذن الله فيما بعد إلى دراسـة تأثير عـدم التحديـد (nondeterminism) في الأوضاع العملية (practical situations) ، ولذلك نحتاج الآن إلى إضافة بعض التعليقات . كما هو الحال دائما فإن عدم التحديد يمكن النظر إليه على أنه اختيار بين بدائل (alternatives) . ويمكننا أن نصور ذلك على أنه شجرة قرارات (a decision tree) [انظر شكل ٢ – ١٥] .



شكل ٢ – ١٥

ويعتمد عرض (width) مثل شجرة التشكيلات (configuration tree) هذه على معامل التَفَرُّع (branching factor) ، وهو عدد الاختيارات المتاحة (available options) في كل حركة (on each move) . وإذا كانت k تشير إلى أقصى تَفَرُّع (maximum branching) ، فإن

$$M = k^n \qquad (1)$$

هو أكبر عدد من التشكيلات التي يمكن أن توجد بعد n حركة (n moves) .

ولأسباب سنذكرها بإذن الله فيما بعد ، من الضروري أن نتناول ببعض التفصيل تعريف قبول اللغات (language acceptance) ، وكذلك نتناول مسألة العضوية / الانتماء (membership issue) .

تعريف ٢ – ٣ :

- يقال إن آلة تيورنج غير المحددة M تقبل (accepts) لغةً ما L إذا تحقق الشرط التالي :

لأي سلسلة w في اللغة L $\forall\, w \in L$ توجد على الأقل تشكيلة واحدة من التشكيلات الممكنة (possible configurations) تقبل السلسلة w . وقد توجد تفرعات (branches) تؤدي إلى تشكيلات غير قابلة للسلسلة (nonaccepting configurations) ، بينما قد تؤدي تفرعات أخرى إلى وضع الآلة في عروة لا نهائية (infinite loop) . ولكن هذه وتلك لا تؤثر على تحقق شرط القبول .

- ويقال إن آلة تيورنج غير المحددة M تتخذ قرارا بشأن لغةٍ ما (تحدِّد لغةً ما) L (decides a language) إذا تحقق الشرط التالي :

لأي سلسلة w في $\forall w \in \Sigma^*$ يوجد مسار (a path) يؤدي إما إلى قبول (acceptance) أو إلى رفض (rejection) .

رابعا : آلة تيورنج الشاملة A Universal Turing Machine

نعرض فيما يلي اعتراضا (objection) على رسالة تيورنج ، ثم نرى كيف يمكننا دحضه بتصميم آلة سنطلق عليها "آلة تيورنج شاملة" . أما الاعتراض فهو : " إن آلة تيورنج كما عَرَّفناها في

تعريف ١ – ١ تُعَدُّ حاسوبا لغرض خاص (a special purpose computer) . وبمجرد تعريف دالة الانتقال δ ، تصبح الآلة مقيَّدة (restricted) لتنفيذ نوع معين واحد من الحسابات (one particular type of computation) . ومن ناحية أخرى فإن الحواسيب الرقمية (digital computers) تُعَدُّ آلات ذوات أغراض عامة (general purpose machines) ويمكن برمجتها لتنفيذ مهام مختلفة في أوقات مختلفة . وبالتالي فإن آلات تيورنج لا يمكن اعتبارها مكافئة للحواسيب الرقمية عامة الأغراض (general purpose digital computers) .

ويمكننا دَفْع هذا الاعتراض ودحضه بتصميم آلة تيورنج قابلة لإعادة البرمجة (reprogrammable Turing machine) ، يُطْلَق عليها : " آلة تيورنج عامة / شاملة " (**a universal Turing machine**) .

تعريف ٢ – ٤

آلة تيورنج الشاملة M_u **(a universal Turing machine)** هي آلة ذاتية الحركة (an automaton) يمكنها إذا أعطيناها كمدخلات : وَصْف (description) أي آلة تيورنج M ، وسلسلة w محاكاة حسابات (computation) الآلة M على السلسلة w .

ولإنشاء / لبناء (construction) مثل هذه الآلة M_u ، فإننا نختار أولا طريقة قياسية لوصف آلات تيورنج . ويمكننا أن نفرض – دون أن تتأثر عمومية (generality) عملية بناء الآلة بهذا الفرض – أن

$$Q = \{q_1, q_2, \ldots, q_n\}$$

حيث q_1 هي الحالة الابتدائية ، و q_2 هي الحالة النهائية الوحيدة ، وأن

$$\Gamma = \{a_1, a_2, \ldots, a_m\}$$

حيث a_1 تمثل الفراغ (blank) . ثم نختار بعد ذلك تشفيرا (an encoding) تُمثَّل فيه الحالة q_1 بالرمز **1** ، والحالة q_2 بالسلسلة **11** ، وهكذا . وأما الرمز **0** فسيُستخدم كفاصل (separator) بين الآحاد **1's** . وباستخدام هذا الاصطلاح (convention) لتعريف الحالة الابتدائية والحالة النهائية والفراغ يمكننا وَصْف أي آلة تيورنج وصفا كاملا (completely described) بالدالة

δ فقط . ويتم تشفير (encoding) دالة الانتقال بناء على هذه الطريقة (scheme) ، حيث الوسطاء (arguments) والنتيجة (result) تكون في متتابعةٍ ما قد تم تحديدها / وصفها (in some prescribed sequence) .

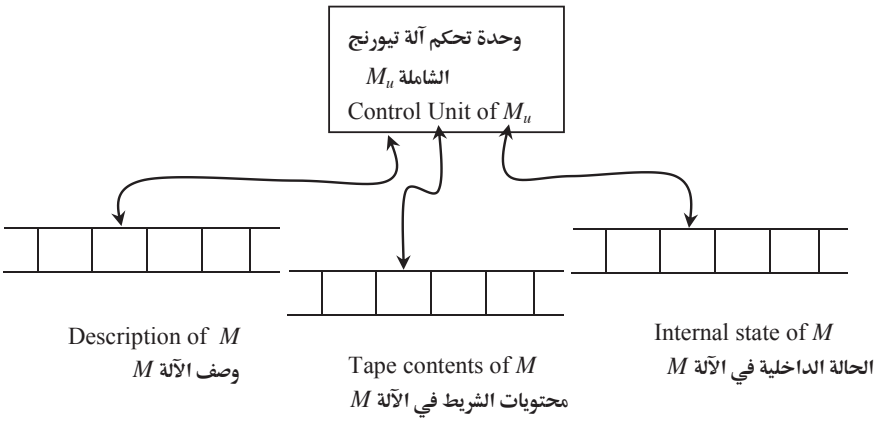
مثال ٢ – ٥ :

الانتقال $\delta(q_1, a_2) = (q_2, a_3, L)$ قد يظهر مثلاً هكذا :

...1 0 1 1 0 1 1 0 1 1 1 0 1 0 ...

وبنـاء على هـذا فإن أي آلة تيـورنج سيكون لها تشفير محدود ((a finite encoding) عبارة عن سلسلة (string) على $\{0, 1\}^+$ ، وإذا أُعطينا أي تشـفير (encoding) لآلة **M** فسيمكننا ترجمة / فك (decoding) هذا التشفير بطريقة وحيدة (uniquely) . وهناك بعض السلاسل التي سوف لا تمثل أي آلة تيورنج (مثلا : السلسلة 00011) ، ولكن سيمكننا بسهولة التعرُّف على هذه السلاسل ، وبالتالي فلن نهتم بها .

وهكذا فإن أي آلة تيـورنج شاملة M_u سيكون لها مجموعة أبجدية مُدخلة (input alphabet) تحتوي على $\{0, 1\}$ ، وبنية آلة متعددة الأشرطة (structure of a multitape machine) ، كما هو موضح في شكل ٢ – ١٦ .



شكل ٢ ـ ١٦

لأي مدخـــلات M, w سيقـــوم الشريـــط tape 1 بالاحتفـاظ بتعريـــف مشفَّر (an encoded definition) للآلة M . وأما الشريط tape 2 فسيحتوي على محتويات شريط الآلة M (tape contents of) ، والشريط tape 3 على الحالة الداخلية للآلة (internal state of) M ، وتقوم آلة تيورنج الشاملة M_u بالنظر أولا إلى محتويات الشريطين 2, 3 لتحديد الهيئة التكوينية / تشكيلة الآلة M (configuration of) . ثم تتجه إلى الشريط tape 1 لترى ماذا ستفعـل الآلة M في هذه التشكيلة . وأخيرا سيتم تعديل (modifying) الشريطين 2, 3 بما يبيِّن / يعكس (reflect) نتيجة الحركة .

ومن الممكن إنشاء (constructing) آلة تيورنج شاملة ولكن العملية ليست مشجِّعة ، ونفضِّل بدلا من ذلك اللجوء إلى فرضية تيورنج (Turing's hypothesis) . ومن الواضح أنه يمكننا التنفيذ (implementation) باستخدام إحدى لغات البرمجة ، وبذلك يكون مثل هذا البرنامج تحقيقا (a realization) لآلة تيورنج الشاملة بهذه اللغة . ولذلك فإننا نتوقع أنه يمكننا أيضا التنفيذ بآلة تيورنج قياسية . وهذا يبرر لنا الادعاء (claiming) بوجود آلة تيورنج تستطيع إذا أُعطيت أي برنامج أن تقوم بتنفيذ الحسابات (carrying out the computations) التي يحددها هذا البرنامج ، وبالتالي فإن تلك الآلة تكون نموذجا صحيحا (a proper model) لحاسوب متعدد الأغراض (a general – purpose computer) .

وتجدر الإشارة إلى أن الملاحظة (observation) أن " أي آلة تيورنج يمكن أن تمثَّل (represented) بسلسلة من الأصفار 0's والآحاد 1's " لها اقتضاءات (implications) / نتائج هامة . ولكن لمناقشة تلك النتائج سنحتاج أولا إلى مراجعة بعض النتائج من نظرية المجموعات (set theory) .

بعض المجموعات تكون محدودة (finite) ، ولكن معظم المجموعات (واللغات) المفيدة والعملية تكون لا نهائية / غير محدودة (infinite) . وبالنسبة لهذه المجموعات اللانهائية فإننا نميِّز ما بين المجموعات القابلة للعَدِّ **(countable)** والمجموعات غير القابلة للعد **(uncountable)** . ويقال لمجموعة إنها قابلة للعد إذا أمكن وضع عناصرها في تقابل واحد لواحد (one – to –one correspondence) مع الأعداد الصحيحة الموجبة . ونعني بذلك أنه يمكن كتابة عناصر المجموعة بترتيبٍ ما ، مثل : $x_1, x_2, x_3, \ldots$ بحيث يكون لكل عنصر في المجموعة مؤشِّر محدود (some finite index) . فمثلا مجموعة جميع الأعداد الصحيحة الزوجية يمكن أن تُكتب بالترتيب $0, 2, 4, \ldots$. ونظراً لأن أي عدد صحيح موجب $2n$ يظهر

في الموضع $n+1$ ، فلذلك تُعَدُّ هذه المجموعة قابلة للعد . ولكن هناك بعض الأمثلة الأكثر تعقيدا والقابلة أيضا للعد ولكنها تبدو غير قابلة للعد .

مثال ٢ – ٦ :

نفرض أن لدينا مجموعة الكسور الاعتيادية (quotients) التي صيغتها p / q ، حيث كل من p, q عدد صحيح موجب . كيف يمكننا ترتيب عناصر هذه المجموعة لِنُبيِّن أنها قابلة للعد ؟

لا يمكننا استخدام المتتابعة $\frac{1}{1}, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \cdots$ لأن كسرا مثل $\frac{2}{3}$ سوف لا يظهر إطلاقا . ولكن هذا لا يعني أن المجموعة غير قابلة للعد . هناك طريقة ذكية لترتيب عناصر المجموعة لبيان أنها قابلة للعد . انظر إلى المخطط (scheme) المبين في الشكل التالي (شكل ٢ – ١٧) ، واكتب العناصر بالترتيب الذي توضحه الأسهم في الشكل ، أي مُتَتَبِّعاً الأسهم .

$$\begin{array}{ccccc}
\frac{1}{1} & \rightarrow & \frac{1}{2} & & \frac{1}{3} \cdots \\
\frac{2}{1} & \swarrow & \frac{2}{2} & \nearrow & \frac{2}{3} \cdots \\
\frac{3}{1} & \nearrow & \frac{3}{2} & \swarrow & \frac{3}{3} \cdots \\
\vdots & \swarrow & \vdots & & \vdots
\end{array}$$

شكل ٢ – ١٧

نحصل من الشكل على المتتابعة :

$$\frac{1}{1}, \frac{1}{2}, \frac{2}{1}, \frac{3}{1}, \frac{2}{2}, \cdots$$

لاحظ هنا أن العنصر $\frac{2}{3}$ يظهر في الموضع السابع ، وأن أي عنصر في المجموعة له موضع ما في هذه المتتابعة . ولذلك فإن هذه المجموعة قابلة للعد .

نرى من هذا المثال أنه يمكننا إثبات أن مجموعةً ما قابلة للعد إذا أمكننا الوصول إلى طريقة لكتابة عناصرها في متتابعةٍ ما . ومثل هذه الطريقة نطلق عليها " طريقة / إجراء تعديد / عَدّ " **(an enumeration procedure)** . ونظرا لأن أي إجراء تعديد / عد هو نوع من عملية ميكانيكية (mechanical process) ، فيمكننا استخدام نموذج آلة تيورنج (Turing machine model) لتعريفه شكليا (formal definition) .

تعريف ٢ – ٥ :

نفرض أن S مجموعة سلاسل (strings) على مجموعةٍ أبجديةٍ ما Σ . إجراء التعديد (an enumeration procedure) للمجموعة S هو آلة تيورنج يمكنها تنفيذ متتابعة الخطوات (sequence of steps) :

$$q_0\square \vdash^* q_s x_1 \# \ s_1 \vdash^* q_s x_2 \# s_2 \ \ldots$$

حيث

$$x_1 \in \Gamma^* - \{\#\}, s_i \in S$$

بطريقة تجعل أي سلسلة s في المجموعة S يتم الحصول عليها (produced) بعدد محدود (finite) من الخطوات . والحالة q_s هي حالة تعني (signifies) الانتماء إلى / العضوية (membership) في المجموعة s ، أي أنه كلما أُدخلت q_s ، فإن السلسلة التي تلي # يجب أن تكون في المجموعة s .

ويلاحظ أنه ليست أي مجموعة تكون قابلة للعد (countable) . وكما سنرى بإذن الله في الفصل القادم هناك بعض المجموعات غير القابلة للعد (uncountable sets) . ولكن أي مجموعة يوجد لها إجراء تعديد تكون قابلة للعد لأن التعديد (enumeration) يعطي المتتابعة المطلوبة .

وإذا أردنا الدِّقَّة في التعبير فإن أي إجراء تعديد لا يمكننا أن نطلق عليه خوارزمية (an algorithm) ، وذلك لأنه سوف لا يتوقف (will not terminate) عندما تكون المجموعة S لا نهائية (infinite) . ومع ذلك فإنه يمكننا اعتباره عملية ذات معنى / مغزى (a

(well – defined لأنه يعطينا نتائج محدَّدة ومُعَرَّفة جيدا ومُتوقَّعة meaningful process) . and predictable results)

مثال ٢ – ٧ :

نفرض أن $\sum = \{a, b, c\}$. يمكننا إثبات أن المجموعة $S = \sum^{+}$ قابلة للعد إذا أمكننا إيجاد إجراء تعديد يعطينا عناصر هذه المجموعة بترتيبٍ ما ، وليكن الترتيب الذي تظهر به في قاموس . إلا أن الترتيب المستخدم في القواميس ليس مناسبا بدون تعديل . ففي القاموس جميع الكلمات التي تبدأ بالحرف a تظهر قبل السلسلة b . ولكن إذا كان عناك عدد لا نهائي من كلمات a ، فلن نصل إطلاقا إلى b ، وهذا يناقض شرط تعريف إجراء التعديد (تعريف ٢ – ٥) أن أي سلسلة معطاة يجب أن نحصل عليها بعد عدد محدود من الخطوات .

وبدلا من ذلك فإنه يمكننا استخدام ترتيب معدَّل (a modified order) نأخذ فيه طول السلسلة (length of the string) كأول معيار (first criterion) ، يليه ترتيب أبجدي لجميع السلاسل متساوية الطول . وهذا إجراء تعديد يعطي المتتابعة .

$$a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, \ldots.$$

ونظرا لأنه ستكون هناك بإذن الله استخدامات عدة لمثل هذا الترتيب فسنطلق عليه الترتيب الصحيح / السليم / المضبوط **(proper order)** . ومن النتائج الهامة لما سبق ذكره أن آلات تيورنج تُعَدُّ قابلة للعد .

نظرية ٢ – ٣ :

مجموعة جميع آلات تيورنج قابلة للعد ، رغم أنها لا نهائية .

البرهان :

يمكننا تشفير (encoding) أي آلة تيورنج باستخدام الصفر **0** والواحد **1** . وباستخدام هذا التشفير نقوم بإنشاء إجراء التعديد التالي :

١) قم بتوليد (generating) السلسلة التالية (next string) في $\{0, 1\}^{+}$ بترتيب صحيح .

٢) تَحَقَّق ما إذا كانت السلسلة المُوَلَّدة تُعَرِّف آلة تيورنج أم لا . فإن كانت تُعَرِّف آلة تيورنج ، فاكتبها على الشريط بالصيغة التي يتطلبها تعريف ٢ – ٥ . وإن كانت لا تُعَرِّف فاهمل السلسلة .

٣) ارجع إلى الخطوة 1.

ونظرا لأن أي آلة تيورنج لها وصف محدود (a finite description) ، فأي آلة معينة (specific machine) سيتم في النهاية (eventually) توليدها بالعملية (process) المذكورة .

* * *

والترتيب الخاص (particular ordering) لآلات تيورنج يعتمد على التشفير الذي نستخدمه . وإذا استخدمنا تشفيرا مختلفا فيجب أن نتوقع ترتيبا مختلفا . ولكن هذا ليس له أي أهمية ، فالترتيب نفسه ليس مهما ، ولكن المهم هو وجود ترتيب ما .

خامسا : الآلات المقيَّدة الخطية Linear Bounded Automata

بينما لا يمكننا توسيع (extending) / زيادة قدرة (power) آلة تيورنج القياسية عن طريق تعقيد (complicating) بنية الشريط (tape structure) ، إلا أنه يمكننا الحد من (limiting) هذه القدرة عن طريق تقييد (restricting) الطريقة التي يمكن أن يُستخدم بها الشريط . وقد رأينا فعلا مثالا لذلك في حالة آلات الدفع لأسفل (pushdown automata) . فيمكننا اعتبار آلة الدفع لأسفل آلة تيورنج غير محدَّدة (nondeterministic) ذات شريط مقيَّد (restricted) بأن يُستخدم كَرَصَّة (like a stack) . ويمكننا كذلك تقييد استخدام الشريط بطرق أخرى ، كأن نسمح مثلا باستخدام جزء محدود فقط (only a finite part) من الشريط كحيِّز للعمل (as work space) . ويمكننا إثبات أن هذا يؤدي إلى (يرجع بنا إلى) الآلات المحدودة (finite automata) . ولكن هناك طريقة لتقييد استخدام الشريط تؤدي إلى حالة ذات أهمية خاصة . وذلك بأن نسمح للآلة أن تَستخدم فقط جزء الشريط الذي تشغله المدخلات (occupied by the input) . وبالتالي يتوفر لنا حيِّز أكبر (more space) لسلاسل مدخلات طويلة (long input signals) من ذلك الذي يكون متاحا لسلاسل قصيرة . وهذه الطريقة تولِّد لنا طبقة أخرى من الآلات يُطلق عليها " الآلات المقيَّدة الخطية " **(linear bounded automata)** ، أو اختصارا آلات **lba** .

والآلة المقيَّدة الخطية – مثل آلة تيورنج القياسية – لها شريط غير مقيَّد (unbounded tape) ، ولكن قَدْر الجزء من الشريط الذي يمكن أن يُستخدم يُعتبر دالة في المدخلات (function of the input) . وبصورة خاصة فإننا نقيِّد الجزء القابل للاستخدام من الشريط

تحديدا على تلك الخلايا التي تشغلها المدخلات(*) . ولتنفيذ ذلك يمكننا وضع المدخلات بين علامتين / رمزين خاصين (two special symbols) : علامة النهاية اليسرى **left-end marker** (" [") ، وعلامة النهاية اليمنى "] " . وبالنسبة لسلسلة مدخلات w ، فإن التشكيلة الابتدائية (initial configuration) لآلة تيورنج تُعطَى بالوَصْف اللحظي (instantaneous description) $q_0\ [w]$. وعلامتا النهاية (end markers) لا يمكن إعادة كتابتهما ، ورأس القراءة والكتابة لا يمكنها التحرك إلى يسار العلامة [ولا إلى يمين العلامة] . وأحيانا نقول إن رأس القراءة والكتابة تقفز وترتد (bounces off) بين علامتي النهاية .

تعريف ٢ – ٦ :

الآلة المقيَّدة الخطية (**linear bounded automaton**) هي آلة تيورنج غير محدَّدة (a nondeterministic Turing machine) $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ [والمعرَّفة في تعريف ٢ – ٢] تخضع للقيد أن المجموعة الأبجدية Σ يجب أن تحتوي على رمزين خاصين [و] (2 special symbols) ، بحيث أن $\delta(q_i, [\ ])$ يمكن أن تحتوي فقط على عناصر صيغتها $(q_j, [, R)$ ، وأن $\delta(q_i,]\ )$ يمكن أن تحتوي فقط على عناصر صيغتها $(q_j,], L)$.

تعريف ٢ – ٧ :

أي سلسلة w يقال إنها تقبلها آلة مقيَّدة خطية إذا وُجدت متتابعة ممكنة من التحركات (a possible sequence of moves)

$$q_0\ [w] \vdash^* [x_1\, q_f\, x_2]$$

لبعض $q_f \in F,\ x_1, x_2 \in \Gamma^*$. واللغة التي تقبلها الآلة المقيَّدة الخطية **lba** هي مجموعة جميع هذه السلاسل التي تقبلها تلك الآلة .

(*) في بعض التعريفات يكون الجزء القابل للاستخدام من الشريط (usable part of the tape) أحد مضاعفات طول المدخلات (a multiple of the input length) ، حيث يمكن أن يعتمد هذا المضاعف (multiple) على اللغة (language) وليس على المدخلات . وهنا سنستخدم نحن طول سلسلة الإدخال بالضبط فقط (وليس مضاعفاً له) ، ولكننا سنسمح بآلات متعددة المسارات (multitrack machines) ، حيث المدخلات تكون على مسار واحد فقط .

ملاحظة :

يُلاحَظ في هذا التعريف (تعريف ٢ – ٧) أن الآلة المقيَّدة الخطية نفترض أنها غير محددة (nondeterministic) . وهذا ليس مجرد فرض لتبسيط الأمور والملاءمة (convenience) ولكنه أساسي (essential) بالنسبة لمناقشة الآلات المقيَّدة الخطية . ورغم أنه يمكننا تعريف الآلات المقيَّدة الخطية المحدَّدة (deterministic) ، إلا أنه ليس معلوما ما إذا كانت هذه الآلات مكافئة لتلك غير المحددة .

مثال ٢ – ٨ : اللغة

$$L = \{a^n b^n c^n : n \geq 1\}$$

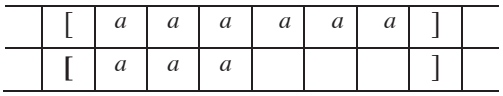
تَقبَلُها آلة مقيَّدة خطية . وهذا ينتج من المناقشة التي ذكرناها في حل مثال ١ – ١٠ . فالحسابات (computation) التي لَخَّصْنا خطواتها هناك لا تتطلب حيِّزا خارج المدخلات الأصلية ، وبالتالي يمكن تنفيذها بآلة مقيَّدة خطية .

مثال ٢ – ٩ :

أوجد آلة مُقَيَّدة خطية تقبل اللغة

$$L = \{a^{n!} : n \geq 0\}$$

إحدى طرق حل هذه المسألة هي أن نقسم عدد الرموز $a's$ على التتابع على 2, 3, 4, … حتى يمكننا إما أن نقبل أو أن نرفض السلسلة . فإن كانت المدخلات موجودة في اللغة L ، فإنه سيتبقى لدينا في النهاية رمز واحد a ، وإن لم تكن في اللغة L فسيظهر لنا في وقتٍ ما باقي غير صفري (a nonzero remainder) . وسنوضح مخططا للحل للإشارة إلى أحد الاقتضاءات (implications) الضمنية لتعريف ٢ – ٦ . فنظراً لأن شريط الآلة المقيدة الخطية قد يكون متعدد المسارات (multitrack) ، فإن المسارات الإضافية يمكن أن تُستخدم كَحَيِّز للعمل (work space) (لإجراء الحسابات) . وفي المسألة الحالية يمكننا استخدام شريط ثنائي المسار (a two - track tape) : المسار الأول يحتوي على عدد الرموز $a's$ المتبقية (left) أثناء عملية القسمة (process of division) ، والمسار الثاني يحتوي على القاسم الحالي (current divisor) [انظر شكل ٢ – ١٨] . والحل الفعلي بسيط : مستخدمين القاسم على المسار الثاني ، نقسم عدد



شكل ٢ ـ ١٨

الرموز $a's$ الموجودة على المسار الأول ، مثلاً بحذف جميع الرموز ما عدا تلك التي عند مضاعفات القاسم (at multiples of the divisor) . وبعد ذلك نزيد القاسم بواحد ، ونستمر في تلك العملية إلى أن نصل إلى إحدى حالتين : إما أن نجد باقيا غير صفري (a nonzero remainder) ، أو أن يتبقى لدينا رمز a وحيد (a single a) .

* * *

المثالان الأخيران يشيران إلى أن الآلات المقيَّدة الخطية أقوى وأكثر قدرة من آلات الدفع لأسفل نظرا لأن أيًّا من اللغتين ليست حرة السياق (context - free) . ولإثبات هذه المقولة / الفرض / التخمين علينا أن نثبت كذلك أن أي لغة حرة السياق يمكن أن تقبلها آلة مقيَّدة خطية [انظر مثلا السؤال ٢ – ١٥] . وليس من السهل تخمين العلاقة بين آلات تيورنج والآلات المقيَّدة الخطية . فالمسائل الشبيهة بتلك التي يعرضها مثال ٢ – ٩ يمكن حلها بآلة مقيَّدة خطية نظراً لتوفر حيِّز للعمل (والتسويد) (scratch space) يتناسب مع (proportional to) طول المدخلات . وفي الحقيقة فإنه من الصعب جدا أن نأتي بلغة معرَّفة تعريفا دقيقا صريحا لا يمكن أن تقبلها أي آلة مقيَّدة خطية . وفي الفصل القادم (الفصل الثالث) سنبيِّن بإذن الله أن طبقة الآلات المقيَّدة الخطية أقل قوة وقدرة (less powerful) من طبقة آلات تيورنج غير المقيَّدة (unrestricted Turing machines) ، ولكن عرض (demonstration) ذلك يتطلب جهدا أكبر .

تمرينات رقم (٢)

أولا : تغييرات طفيفة على آلة تيورنج الأساسية

٢ – ١) نفرض أن لدينا آلـة تيورنـج يمكنها عند أي حركة معينة (any particular move) إما أن تُغَيِّر رمز الشريط أو أن تُحَرِّك رأس القراءة والكتابة ولكن لا تُنفِّذ العمليتين معا .

أ) اكتب تعريفا رسميا / شكليا (a formal definition) لهذه الآلة .

ب) اثبت أن طبقة هذه الآلات مكافئة لطبقة آلات تيورنج القياسية .

٢ – ٢) آلة تيورنج غير الماحية ((a nonerasing Turing machine) هي آلة لا يمكنها تغيير رمز غير فارغ (a nonblank symbol) إلى فراغ (a blank) . ويمكن الوصول إلى ذلك بوضع القيد (restriction) التالي :

إذا كانت

$$\delta\,(q_i, a) = (q_j, \square, L \text{ or } R)$$

فإن a يجب أن تكون □ . اثبت أننا لا نفقد أي عمومية (generality) بوضع مثل هذا القيد .

٢ – ٣) نفرض أننا قد وضعنا قيدا على آلة تيورنج بأن عليها أن تكتب دائما رمزا مختلفا عن الرمز الذي قرأته ، أي أنه إذا كانت

$$\delta\,(q_i, a) = (q_j, b, L \text{ or } R)$$

فإن الرمزين a, b يجب أن يكونا مختلفين . هل هذا القيد / التحديد (limitation) يقلِّل من قدرة الآلة (power of the automaton) ؟

٢ – ٤) نفرض أن لدينا آلة تيورنج ذات عملية " اتخاذ قرار " مختلفة (a different decision process) تتم فيها الانتقالات (transitions) إذا لم يكن رمز الشريط الحالي (current tape symbol) أحد عناصر مجموعة معيَّنة (a specified set) . فمثلا الانتقال

$$\delta\,(q_i, \{a, b\}) = (q_j, , c, R)$$

سيسمح بالحركة المشار إليها إذا لم يكن رمز الشريط الحالي a أو b. اكتب هذا المفهوم بصيغة رسمية / شكلية (formalize this concept)، واثبت أن هذا التعديل (modification) مكافئ لآلة تيورنج قياسية.

ثانيا : آلات تيورنج ذوات تخزين أكثر تعقيدا

٢ – ٥) آلة تيورنج متعددة الرؤوس (a multihead Turing machine) يمكن النظر إليها على أنها آلة تيورنج ذات شريط واحد فقط (a single tape)، ووحدة تحكم واحدة فقط (a single control unit)، ولكن عدة رؤوس قراءة وكتابة مستقلة (multiple independent read – write heads). اكتب تعريفا رسميا / شكليا لآلة تيورنج متعددة الرؤوس، ثم وَضِّح كيف يمكن محاكاة مثل هذه الآلة بآلة تيورنج قياسية.

٢ – ٦) تُعَرَّف " آلة طابور ذاتية الحركة " (**a queue automaton**) بأنها آلة ذاتية الحركة أداة التخزين المؤقت (temporary storage) فيها عبارة عن طابور (queue). نفرض أن هذه الآلة موصولة على الخط (an on-line machine)، أي أنها لا تحتوي على ملف إدخال (input file)، والسلسلة (string) المطلوب تشغيلها (to be processed) موضوعة في الطابور قبل بداية إجراء الحسابات. اكتب تعريفا شكليا لهذه الآلة، ثم ادرس قدرتها (power) بالنسبة لآلات تيورنج.

٢ – ٧) اثبت أن أي عملية حسابية يمكن إجراؤها بآلة تيورنج قياسية يمكن أيضا إجراؤها بآلة متعددة الأشرطة ذات اختيار البقاء في الموضع (a multitape machine with a stay - option) وحالتين على الأكثر (at most two states).

ثالثا : آلات تيورنج غير المحدَّدة

٢ – ٨) اكتب برنامجا لآلة تيورنج غير محدَّدة تقبل اللغة

$$L = \{ww : w \in \{a, b\}^+\}$$

قارن حلك بحل محدَّد (a deterministic solution).

٢ – ٩) صمِّم آلة تيورنج غير محدَّدة تقبل اللغة

$$L = \{a^n : n \text{ is not a prime number } (n \text{ ليس عددا أوَّليا})\}$$

٢ – ١٠) الآلة ذاتية الحركة ذات الرَّصَّتين (a two-stack automaton) هي آلة ذاتية الحركة غير محددة ذات دفع لأسفل وذات رصتين مستقلتين (a nondeterministic pushdown automaton with two independent stacks) . ولتعريف هذه الآلة M فإننا نُجري تعديلا على تعريف آلة القبول غير المحدَّدة ذات الدفع لأسفل (nondeterministic pushdown accepter) **npda** بحيث تصبح دالة الانتقال δ كما هو مبين في التعريف التالي :

$$M = (Q, \sum, \Gamma, \delta, q_0, z, F)$$

حيث

Q : مجموعة محدودة من الحالات الداخلية لوحدة التحكم .

$\sum$: المجموعة الأبجدية للإدخال .

Γ : مجموعة محدودة من الرموز تُدعَى : المجموعة الأبجدية للرصة (**stack alphabet**) .

δ : دالة الانتقال

$$\delta : Q \times (\sum U \{\lambda\}) \times \Gamma \times \Gamma \rightarrow$$

مجموعات جزئية محدودة من finite subsets of $Q \times \Gamma^* \times \Gamma^*$

$q_0 \in Q$: الحالة الابتدائية لوحدة التحكم .

$z \in Q$: الرمز الابتدائي للرصة (stack start symbol) .

$F \subseteq Q$: مجموعة الحالات النهائية .

أي حركة للآلة تعتمد على قمتي الرصتين وينتج عنها قيم جديدة تُدفع (pushed on) على هاتين الرصتين . اثبت أن طبقة الآلات ذاتية الحركة ذوات الرصتين مكافئة لطبقة آلات تيورنج .

رابعا : آلة تيورنج الشاملة

٢ – ١١) اكتب مخططا لبرنامج آلة تيورنج يَعُدُّ / يُعَدِّد (enumerates) المجموعة $\{0,1\}^+$ بترتيب صحيح / سليم (in proper order) .

٢ – ١٢) نفرض أن S_1, S_2 مجموعتان قابلتان للعد (countable sets) . اثبت أن المجموعتين $S_1 \cup S_2$ و $S_1 \times S_2$ ستكونان أيضا قابلتين للعد .

خامسا : الآلات المقيَّدة الخطية

٢ – ١٣) أوجد حلا لمثال ٢ – ٩ لا يتطلب وجود مسار ثاني (a second track) لإجراء الحسابات ومحاولات الوصول إلى الحل .

٢ – ١٤) أوجد آلة مقيَّدة خطية لقبول اللغة

$$L = \{w^n : w \in \{a, b\}^+, n \geq 1\}$$

الفصل الثالث

التسلسل الهرمي للغات الشكلية والآلات ذاتية الحركة
A Hierarchy of Formal Languages and Automata

نوجّه اهتمامنا في هذا الفصل إلى دراسة اللغات الشكلية وهدفنا المباشر هو دراسة اللغات المرتبطة بآلات تيورنج وبعض القيود المفروضة عليها . ونظرا لأن آلات تيورنج يمكنها إجراء أي نوع من الحسابات الخوارزمية (algorithmic computation) ، فإننا نتوقع أن نجد أن عائلة اللغات المرتبطة بها كبيرة جدا . فهي لا تشمل اللغات المنتظمة واللغات حرة السياق (regular and context - free languages) فحسب ، وإنما تشمل أيضا الأمثلة العديدة التي قابلناها والتي تقع خارج نطاق هاتين العائلتين . والسؤال غير البسيط هو : هل هناك أي لغة لا تقبلها أي آلة من آلات تيورنج ؟ وسنجيب بإذن الله على هذا السؤال أولا ببيان أنه توجد لغات أكثر من آلات تيورنج ، بحيث أنه توجد قطعا بعض اللغات التي لا تقابلها (لا توجد لها) أي آلات تيورنج . والبرهان قصير وممتاز ولكنه ليس بنائياً (ليس إنشائيا) (nonconstructive) ، ويُلْقِي ضوءا باهتا على المسألة . ولهذا السبب فإننا سندعم وجود لغات لا تتعَرَّف عليها آلات تيورنج عن طريق إعطاء المزيد من الأمثلة الصريحة التي تسمح لنا فعليا بالتعرُّف على وتحديد إحدى هذه اللغات . وهناك طريق آخر لدراسة هذه المسألة وهو أن ننظر إلى العلاقة بين آلات تيورنج وأنواع معينة من القواعد (grammars) ، ونوجد رابطا (a connection) بين هذه القواعد من ناحية والقواعد المنتظمة والقواعد حرة السياق (regular and context free grammars) من ناحية أخرى . وهذا يؤدي إلى تسلسل هرمي للقواعد ، ومن خلاله إلى طريقة لتصنيف (classifying) عائلات اللغات (language families) . وهناك بعض مخططات نظريات المجموعات التي توضح العلاقات بين عائلات اللغات المختلفة .

وكي نكون أكثر دقةً وتحديدا فإن الكثير من الحجج (arguments) التي سنوردها في هذا الفصل تكون صالحة (valid) فقط بالنسبة للغات التي لا تحتوي على السلسلة الفارغة / الخاوية (empty string) . وهذا القيد ينشأ من حقيقة أن آلات تيورنج – كما عَرَّفناها – لا يمكن أن تقبل السلسلة الخاوية . وكي نتجنب ضرورة إعادة صياغة هذا التعريف أو ضرورة إضافة "رمز تنازل " مكرَّر (a repeated disclaimer) ، فإننا نفترض ضمناً أن اللغات التي نتناولها في هذا الفصل لا تحتوي على السلسلة الخاوية λ ، ما لم يُنص على غير ذلك .ومن السهل إعادة ذكر كل ما سنتناوله ونصل إليه بحيث تكون هناك سلسلة خاوية λ ، ولكننا سنترك ذلك للقارئ الكريم .

أولا : اللغات الارتدادية واللغات القابلة للتعديد ارتداديا
Recursive and Recursively Enumerable Languages

نبدأ فيما يلي ببعض المصطلحات الخاصة باللغات المرتبطة بآلات تيورنج . ويجب هنا أن نميِّز بين مجموعة اللغات التي توجد لها آلة تيورنج تقبلها (an accepting Turing machine) ، ومجموعة اللغات التي توجد لها خوارزمية عضوية / انتماء (a membership algorithm) . ونظرا لأن أي آلة تيورنج لا تتوقف (halt) بالضرورة بالنسبة للمدخلات (input) التي لا تقبلها ، فإن المجموعة الأولى لا تقتضي (imply) المجموعة الثانية .

تعريف ٣ – ١ :

أي لغة L يقال إنها قابلة للتعديد ارتداديا **(recursively enumerable)** إذا وُجدت آلة تيورنج تقبلها .

هذا التعريف يقتضي فقط أنه توجد آلة تيورنج M ، بحيث أنه $\forall w \in L$

$$q_0 w \vdash^*_M x_1 \, q_f \, x_2$$

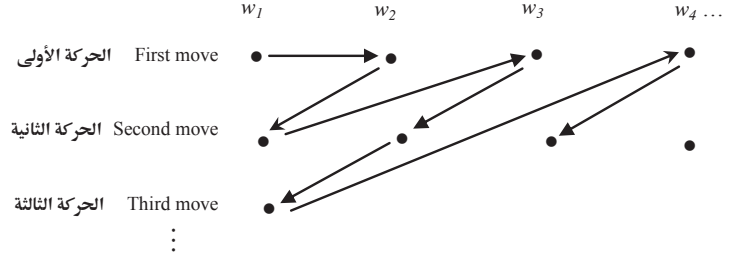
حيث q_f حالة نهائية . ويلاحظ أن هذا التعريف لا يذكر شيئا عما يحدث في حالة ما إذا كانت السلسة w لا تنتمي إلى اللغة L . في هذه الحالة قد تتوقف الآلة في حالة غير نهائية ، وقد لا تتوقف إطلاقا وتدخل في عروة لا نهائية . ويمكننا أن نطلب من الآلة أن تخبرنا ما إذا كانت أي مدخلات معطاة تنتمي إلى لغتها أم لا .

تعريف ٣ – ٢ :

أي لغة L على المجموعة الأبجدية Σ يقال إنها ارتدادية **(recursive)** إذا وُجدت آلة تيورنج M تقبل اللغة L وتتوقف لكل سلسلة w في Σ^+ . وبأسلوب آخر فإن أي لغة تكون ارتدادية إذا وفقط إذا وُجدت لها خوارزمية عضوية / انتماء (a membership algorithm) .

إذا كانت لغةٌ ما ارتـــدادية ، فإنــه يوجـد إجـــراء تعديـد يسهـــل إنـشاؤه (easily constructed enumeration procedure) . نفـرض أن M آلة تيـورنج تحدد العضوية في / الانتماء لـ (membership) لغةٍ ارتدادية L . ننشئ أولا آلة تيورنج أخـــرى – ولتكـــن $\hat{M}$ – تُولِّد جميع السلاسل في Σ^+ بترتيب صحيح / سليم (proper order) ، ولتكن مثلا $w_1, w_2, \ldots$. وعندما يتم توليد هذه السلاسل ، فإنها تصبح مُدخلات الآلة M التي يتم تعديلها (modified) بحيث لا تكتب على شريطها إلا السلاسل فقط الموجودة في اللغة L .

وأما أنه يوجد أيضا إجراء تعديد لكل لغة قابلة للتعديد ارتداديا فَأَمْرٌ لا يُرى بالسهولة نفسها . فنحن لا نستطيع استخدام الحجة (argument) السابقة كما هي ، وذلك لأنه إذا لم تكن سلسلةٌ ما w_j في اللغة L ، فإن الآلة M عندما نبدأها بالسلسلة w_j على شريطها قد لا تتوقف إطلاقا ، وبالتالي لا تصل إطلاقا إلى السلاسل في اللغة L التي تلي السلسلة w_j في التعديد (enumeration) . وكـي نتأكـد ونضـمن ألا يحـدث هـذا فـإن الحسـابات (computation) تُجرى بطريقة مختلفة . حيث نجعل أولا الآلة $\hat{M}$ تولِّد السلسلة w_1 ، ونجعل الآلة M تُنَفِّذ حركة (move) واحدة عليها . ثم نجعل الآلة $\hat{M}$ تولِّد السلسلة w_2 ، ونجعل الآلة M تنفذ حركة واحدة على السلسلة w_2 ، تليها الحركة الثانية على السلسلة w_1 ، وبعد ذلك نولِّد السلسلة w_3 ، وننفذ خطوة (step) واحدة على السلسلة w_3 ، والخطوة الثانية على السلسلة w_2 ، والخطوة الثالثة على السلسلة w_1 ، وهكذا . ويبيِّن شكل ٣ – ١ ترتيب إجراء هذه الخطوات . ومن ذلك يتضح أن الآلة M لن تدخل إطلاقا في عروة لا نهائية . ونظرا لأن أي سلسلة $w \in L$ ستولِّدها الآلة $\hat{M}$ وستقبلها الآلة M بعدد محدود من الخطوات ، فلذلك أي سلسلة في اللغة L ستولِّدها في النهاية الآلة M .



شكل ٣ – ١

ومن السهل أن نرى أن أي لغة يوجد لها إجراء تعديد تكون قابلة للتعديد ارتداديا . فببساطة نقارن بين سلسلة المدخلات المعطاة والسلاسل المتتابعة (successive strings) التي يولدها إجراء التعديد . فإذا كانت $w \in L$ فسنحصل في النهاية على توافق (match) ، ويمكن للعملية أن تُنْهَى (terminated) .

ويلاحظ أن تعريفي ٣ – ١ و ٣ – ٢ لا يلقيان نظرة ثاقبة على طبيعة كل من اللغات الارتدادية واللغات القابلة للتعديد ارتداديا . فهذان التعريفان يعطيان أسماء لعائلات اللغات المرتبطة بآلات تيورنج ، ولكنهما لا يلقيان أي ضوء على طبيعة اللغات الممثلة في هذه العائلات . كما أنهما لا يخبرانا كثيرا عن العلاقات بين هذه اللغات أو عن الارتباط بينها وبين عائلات اللغات التي درسناها سابقا . ولذلك فإننا نواجه على الفور أسئلة مثل : هل توجد لغات قابلة للتعديد ارتداديا ولكنها ليست ارتدادية ؟ وهل توجد لغات قابلة للوصف نوعاً ما (describable somehow) ليست قابلة للتعديد ارتداديا ؟ ورغم أننا سنكون قادرين على إعطاء بعض الإجابات على هذين السؤالين ، إلا أننا سنكون غير قادرين على إعطاء أمثلة صريحة (explicit) توضح هذين السؤالين ، وخاصة السؤال الأخير .

اللغات غير القابلة للتعديد ارتداديا
Languages that are Not Recursively Enumerable

يمكننا إثبات وجود لغات غير قابلة للتعديد ارتداديا بطرق عديدة . إحدى هذه الطرق – وهي الطريقة التي سنتبعها بإذن الله – قصيرة جدا وتستخدم نتيجة أساسية في الرياضيات .

نظرية ٣ – ١ :

إذا كانت S مجموعة لا نهائية قابلة للعد (infinite countable set) ، فإن مجموعة قوى هذه المجموعة 2^S (its powerset) غير قابلة للعد (not countable) .

البرهان :

نفرض أن $S = \{s_1, s_2, s_3, \ldots\}$. أيُّ عنصر t من عناصر المجموعة 2^S سيمكننا تمثيله بمتتابعة (sequence) من الأصفار 0's والآحاد 1's ، حيث يوجد واحد 1 في الموضع i إذا وفقط إذا كان s_i في المجموعة t . فمثلا المجموعة $\{ s_2, s_3, s_6 \}$ تُمثَّل هكذا : 01100100... ، بينما المجموعة $\{ s_1, s_3, s_5 \}$ تُمثَّل هكذا : 10101... . ومن الواضح أن

أي عنصر في المجموعة 2^S يمكن تمثيله بمثل هذه المتتابعة ، وأن أي متتابعة كهذه تمثِّل عنصرا وحيدا (a unique element) من عناصر 2^S .

سنبرهن بالتناقض (by contradiction) أن المجموعة 2^S غير قابلة للعد :

نفرض أن المجموعة 2^S قابلة للعد . بالتالي فإن عناصرها يمكن أن تُكتب بترتيبٍ ما ، وليكن $t_1, t_2, \ldots$ ، ويمكننا وضع هذه العناصر في جدول ، كما هو مبين في الشكل التالي (شكل ٣ – ٢) . وفي هذا الجدول نأخذ العناصر القطرية وهي العناصر الموجودة في القطر الرئيسي (main diagonal) ، ونكمِّل (complement) كل عنصر ، أي نستبدل بالواحد 1 صفرا 0 ، وبالصفر 0 واحداً 1 . وفي المثال المبين في شكل ٣ – ٢ العناصر القطرية هي ... 1100 ، وبالتالي نستبدل بها ... 0011 . والمتتابعة الجديدة الناتجة عبر القطر تمثِّل عنصرا ما من عناصر المجموعة 2^S ، وليكن t_i لقيمةٍ ما i . ولكنه لا يمكن أن يكون t_1 لأنه يختلف عن t_1 من خلال s_1 . وللسبب نفسه لا يمكن أن يكون t_2 أو t_3 أو أي عنصر آخر في التعديد (enumeration) . أي أننا وصلنا إلى تناقض ، ما يعني أن الفرض أن المجموعة 2^S قابلة للعد غير صحيح .

t_1	(1)	0	0	0	0	. . .
t_2	1	(1)	0	0	0	. . .
t_3	1	1	(0)	1	0	. . .
t_4	1	1	0	(0)	1	. . .
⋮				⋱		

شكل ٣ – ٢

* * *

نظرا لأن هذا النوع من الحجج (arguments) والبراهين يشتمل على معالجة (manipulation) العناصر القطرية في جدول ، فلذلك يُطْلَق على مثل هذه الطريقة " الإقطار " (diagonalization) . وتُعزَى هذه الطريقة للعالم الرياضي " كانتور " (G. F. Cantor) الذي استخدمها لبيان أن مجموعة الأعداد الحقيقية غير قابلة للعد . وبإذن الله في الفصول التالية سنرى برهانا مشابها في عدة سياقات (contexts) .

وكنتيجة مباشرة للنتيجة السابقة يمكننا بيان أنه عموما توجد آلات تيورنج أقل عددا من اللغات الموجودة ، بحيث أنه توجد حتما بعض اللغات غير القابلة للتعديد ارتداديا .

نظرية ٣ – ٢ :

لأي مجموعة أبجدية غير خالية Σ توجد لغات غير قابلة للتعديد ارتداديا .

البرهان :

أي لغة هي مجموعة جزئية من Σ^* ، وأي مجموعة جزئية من Σ^* هي لغة . ولذلك فإن مجموعة جميع اللغات هي بالضبط 2^{Σ^*} . ونظراً لأن المجموعة Σ^* لانهائية ، فلذلك نظرية ٣ – ١ تخبرنا أن مجموعة جميع اللغات على المجموعة الأبجدية Σ ليست قابلة للعد (not countable) . ولكن مجموعة جميع آلات تيورنج يمكن تعديدها (enumerated) ، وبالتالي فإن مجموعة جميع اللغات القابلة للتعديد ارتداديا قابلة للعد (countable) . وهذا يعني حتماً [باستخدام الفرضية التالية التي نترك إثباتها للقارئ الكريم] وجود بعض اللغات على Σ غير القابلة للتعديد ارتداديا .

فرضية ٣ – ١ :

إذا كانت S_1 مجموعة قابلة للعد (countable) ، و S_2 مجموعة غير قابلة للعد (not countable) ، وكانت $S_1 \subset S_2$ ، فإن المجموعة S_2 يجب أن تحتوي على عدد لا نهائي من العناصر التي لا تنتمي إلى المجموعة S_1 .

ملاحظة :

برهان النظرية السابقة (نظرية ٣ – ٢) رغم أنه قصير وبسيط إلا أنه غير بنائي / غير إنشائي (non constructive) إطلاقا . وكذلك رغم أنه يخبرنا عن وجود بعض اللغات غير القابلة للتعديد ارتداديا ، ولكنه لا يعطينا أي إحساس إطلاقا بشكل هذه اللغات . وفي مجموعة النتائج التالية نحاول الوصول إلى الاستنتاج النهائي بصورة أكثر صراحة .

اللغة غير القابلة للتعديد ارتداديا

A Language that is Not Recursively Enumerable

نظرا لأنه إذا أمكن وصف أي لغة بصيغة خوارزمية مباشرة (described in a direct algorithmic fashion) فإنه يمكن قبولها بآلة من آلات تيورنج ، وبالتالي فإن هذه اللغة تكون

قابلة للتعديد ارتداديا ، ولذلك فإن وصف أي لغة غير قابلة للتعديد ارتداديا يجب أن يكون غير مباشر (indirect) . ومع ذلك فإنه ممكن (possible) . والبرهان يشتمل على تعديل / تغيير (variation) في طريقة الإقطار (diagonalization) .

نظرية ٣ – ٣ :

توجد لغة قابلة للتعديد ارتداديا (a recursively enumerable language) مكمِّلتها (complement) ليست قابلة للتعديد ارتداديا .

البرهان :

• نفـرض أن $\Sigma = \{ a \}$. ونأخـذ في الاعتبار مجموعة جميع آلات تيورنج حيث Σ هي المجموعـة الأبجديـة للمدخلات . مجموعة جميـع آلات تيورنج هـذه قابلـة للعـد (countable) وذلك بنظرية ٢ – ٣ . وبالتالي فيمكننا أن نُلْحِق (associate) ترتيبا (an order) $M_1, M_2, \ldots$ لهذه العناصر . لكل آلة تيورنج M_i توجد لغة قابلة للتعديد ارتداديا $L(M_i)$ مرتبطة بها . وبالعكس : لكل لغة قابلة للتعديد ارتداديا على Σ توجد آلة تيورنج ما تقبل اللغة .

• والآن نأخذ في الاعتبار لغة جديدة L تُعَرَّف كما يلي : لكل $i \geq 1$ تكون السلسلة a^i في اللغة L إذا وفقط إذا كان $a^i \in L(M_i)$. من الواضح أن اللغة L مُعَرَّفة تعريفا جيدا (well defined) نظرا لأن العبارة $a^i \in L(M_i)$ ، وبالتالي العبارة $a^i \in L$ يجب أن تكون إما صحيحة / صادقة (true) أو خاطئة / كاذبة (false) .

• الخطوة التالية هي أن نأخذ في الاعتبار مكمِّلة اللغة L :

$$\overline{L} = \{a^i : a^i \notin L(M_i)\} \qquad (1)$$

والتي هي أيضا مُعَرَّفة تعريفا جيدا ولكنها – كما سنرى – ليست قابلة للتعديد ارتداديا . وسنثبت ذلك بالتناقض (by contradiction) :

نفرض أن اللغة $\overline{L}$ قابلة للتعديد ارتداديا . لذلك يجب أن توجد آلة تيورنج ما ولتكن M_k بحيث أن

$$\overline{L} = L(M_k) \qquad (2)$$

نأخذ في الاعتبار السلسلة a^k . هل هي عنصر في اللغة L أم في اللغة $\overline{L}$ ؟
نفرض أن $a^k \in \overline{L}$. باستخدام العلاقة (2) ، هذا يقتضي

$$a^k \in L(M_k)$$

ولكن العلاقة (1) تقتضي الآن

$$a^k \notin \overline{L}$$

ومن ناحية أخرى إذا فرضنا أن $a^k \in L$ ، فهذا يعني أن $a^k \notin \overline{L}$. وباستخدام العلاقة (2) فإن هذا يقتضي

$$a^k \notin L(M_k)$$

ولكننا عندئذ إذا استخدمنا العلاقة (1) فإننا نحصل على

$$a^k \in \overline{L}$$

أي أنه في أي من الحالتين نصل إلى تناقض ، وبالتالي نستنتج أن فَرْضَنا الأصلي أن $\overline{L}$ قابلة للتعديد ارتداديا فرض خاطئ .

ولتكملة برهان النظرية حسب منطوقها / نَصِّها المذكور علينا أن نثبت أيضا أن L قابلة للتعديد ارتداديا . ولإثبات ذلك يمكننا استخدام إجراء التعديد المعلوم لآلات تيورنج . إذا أُعطينا a^i ، فإننا نجد أولا قيمة i عن طريق حساب عدد الرموز $a's$. ثم نستخدم إجراء التعديد لآلات تيورنج ، وذلك لإيجاد M_i . وأخيرا نعطِي وَصْفَها وكذلك a^i لآلة تيورنج شاملة M_u (a universal Turing machine) تقوم بمحاكاة (simulating) عمل الآلة M (action) على a^i . فإن كانت a^i موجودة في اللغة L ، فإن العمليات الحسابية (computation) التي تقوم بها الآلة M_u ستتوقف في النهاية . والتأثير المشترك لهذا هو آلة تيورنج تقبل كل سلسلة $a^i \in L$. وبناءً على هذا وعلى تعريف ٣ – ١ نستنتج أن اللغة L قابلة للتعديد ارتداديا .

* * *

برهان هذه النظرية يعرض بوضوح (من خلال التعريف ٣ – ١) لغةً مُعَرَّفة تعريفا جيدا وليست قابلة للتعديد ارتداديا . وليس معنى هذا أنه يوجد تفسير بديهي سهل للغة L وسيكون من الصعب أن نعرض أكثر من أعداد قليلة بسيطة من هذه اللغة . ومع ذلك فإن $\overline{L}$ مُعَرَّفة تعريفا صحيحا / سليما (properly defined) .

لغة قابلة للتعديد ارتداديا ولكنها ليست ارتدادية
A Language that is Recursively Enumerable but Not Recursive

والآن نبيِّن فيما يلي أنه توجد بعض اللغات القابلة للتعديد ارتداديا ولكنها ليست ارتدادية . ومرة أخرى نحتاج لتنفيذ ذلك بطريقة التفافية . ونبدأ بتقرير نتيجة جانبية .

نظرية ٣ – ٤ :

إذا كانت لغةٌ ما L ومكمِّلتها اللغة $\overline{L}$ قابلتين للتعديد ارتداديا ، فإن اللغتين ارتداديتان . وإذا كانت اللغة L ارتدادية ، فإن مكملتها اللغة $\overline{L}$ ارتدادية أيضا ، وبالتالي فاللغتان قابلتان للتعديد ارتداديا .

البرهان :

إذا كانت اللغتان L و $\overline{L}$ كلتاهما قابلتين للتعديد ارتداديا ، فإنه توجد آلتا تيورنج M و $\hat{M}$ تعملان كإجرائي تعديد (enumeration procedures) للغتين L و $\overline{L}$ على الترتيب . الآلة الأولى M ستعطِي السلاسل $w_1, w_2, \ldots$ (produce) في اللغة L . والآلة الثانية $\hat{M}$ ستعطِي السلاسل $\hat{w}_1, \hat{w}_2, \ldots$ في اللغة $\overline{L}$. نفرض الآن أننا قد أُعطينا أي سلسلة $w \in \Sigma^+$. نجعل أولا الآلة M تولد السلسلة w_1 ، ونقارنها بالسلسلة w . فإن لم تكونا السلسلة نفسها ، فإننا نجعل الآلة $\hat{M}$ تولِّد السلسلة $\hat{w}_1$ ، ونقارن مرة أخرى . وإذا احتجنا للاستمرار فالخطوة التالية هي أن نجعل الآلة M تولِّد السلسلة w_2 ، ثم تولِّد الآلة $\hat{M}$ السلسلة $\hat{w}_2$ ، وهكذا . وأي سلسلة $w \in \Sigma^+$ ستولِّدها إما الآلة M أو الآلة $\hat{M}$ ، وبالتالي فسنحصل على توافق (a match) في النهاية . فإن كانت السلسلة المتوافقة (matching string) قد أنتجتها الآلة M ، فإن السلسلة w تنتمي إلى اللغة L ، وإلا فإنها تنتمي إلى اللغة $\overline{L}$. وهذه العملية

(process) هي خوارزمية عضوية / انتماء (a membership algorithm) لكل من L و $\overline{L}$ ، وبالتالي فكل منهما لغة ارتدادية .

وبالنسبة للاتجاه العكسي : نفرض أن L لغة ارتدادية . هذا يعني أنه توجد لها خوارزمية عضوية . ولكن هذه تصبح خوارزمية عضوية للغة $\overline{L}$ بتكميل استنتاجها (complementing its conclusion) ببساطة . ولذلك فإن $\overline{L}$ ارتدادية . ونظراً لأن أي لغة ارتدادية هي لغة قابلة للتعديد ارتداديا ، لذلك يكتمل البرهان .

* * *

مما سبق نستنتج مباشرة أن عائلة اللغات القابلة للتعديد ارتداديا وعائلة اللغات الارتدادية ليستا متطابقتين . فاللغة L المذكورة في نظرية ٣ – ٣ موجودة في العائلة الأولى ولكنها غير موجودة في العائلة الثانية .

نظرية ٣ – ٥ :

توجد لغة قابلة للتعديد ارتداديا وليست ارتدادية . أي أن عائلة اللغات الارتدادية مجموعة جزئية صحيحة (a proper subset) من عائلة اللغات القابلة للتعديد ارتداديا .

البرهان :

نأخذ في اعتبارنا اللغة L المشار إليها في نظرية ٣ – ٣ . هذه اللغة قابلة للتعديد ارتداديا ، ولكن مكمِّلتها ليست كذلك . ولذلك فبنظرية ٣ – ٤ هذه اللغة L ليست ارتدادية ، وبذلك فهي تعطينا المثال الذي نبحث عنه .

* * *

مما سبق نرى أنه توجد فعلا لغات مُعَرَّفة تعريفا جيدا (well-defined languages) لا نستطيع أن ننشئ لها خوارزمية عضوية / انتماء (a membership algorithm) .

ثانيا : القواعد غير المقيَّدة Unrestricted Grammars

لدراسة العلاقة بين اللغات القابلة للتعديد ارتداديا والقواعد نعود للتعريف العام للقاعدة الذي يسمح لقواعد الإنتاج (production rules) أن تأخذ أي صيغة دون فرض أي قيود عليها ، وحينئذ نحصل على قواعد غير مقيَّدة .

تعريف ٣ – ٣ :

القاعدة $G = (V, T, S, P)$ يُطْلَق عليها قاعدة غير مقيَّدة (**unrestricted grammar**) إذا كانت جميع القواعد صيغتها

$$u \to v$$

حيث $u \in (V \cup T)^+; \quad v \in (V \cup T)^*$

في أي قاعدة غير مقيَّدة لا تُفْرَض أساسا أي شروط على الإنتاجات . قمن الممكن أن يكون هناك أي عدد من المتغيرات (variables) والطرفيات (terminals) في الطرف الأيسر أو الطرف الأيمن ، وكذلك يمكن أن تظهر بأي ترتيب . وهناك قيد واحد فقط وهو أن السلسلة الخاوية λ لا يُسمح بظهورها في الطرف الأيسر في أي إنتاج (production) .

وكما سنرى بإذن الله فإن القواعد غير المقيَّدة تُعَدُّ أقوى بكثير من القواعد المقيَّدة مثل القواعد المنتظمة والقواعد حرة السياق . وفي الحقيقة فإن القواعد غير المقيَّدة تقابل (correspond to) أكبر عائلة من اللغات ، وبالتالي يمكننا أن نأمل بالتعَرُّف عليها (recognize) بوسائل ميكانيكية (mechanical means) . أي أن القواعد غير المقيَّدة تولِّد بالضبط عائلة اللغات القابلة للتعديد ارتداديا . وسنبيِّن ذلك في جزئين : الأول مباشر ، والثاني يشتمل على عملية بناء / إنشاء (construction) طويلة .

نظرية ٣ – ٦ :

أي لغة تولِّدها قاعدة غير مقيَّدة هي لغة قابلة للتعديد ارتداديا .

البرهان :

القاعدة تعَرَّف فعليا إجراءً لتعديد جميع السلاسل في اللغة بطريقة منظمة (systematically) . فمثلا يمكننا أن نحدد قائمة بجميع السلاسل w في اللغة L بحيث أن

$$S \Rightarrow w$$

أي أن السلسلة w تُشتق بخطوة واحدة . ونظرا لأن مجموعة إنتاجات القاعدة محدودة (finite) ، فسيكون هناك عدد محدود من هذه السلاسل . وبعد ذلك نحدد قائمة جميع السلاسل w في اللغة L التي يمكن أن تُشتق بخطوتين

$$S \Rightarrow x \Rightarrow w$$

وهكذا . ويمكننا محاكاة (simulation) هذه الاشتقاقات (derivations) على آلة تيورنج ، وبالتالي يكون لدينا إجراء تعديد للغة . وبناءً عليه فاللغة قابلة للتعديد ارتداديا .

* * *

هذا الجزء من المقابلة (correspondence) بين اللغات القابلة للتعديد ارتداديا والقواعد غير المقيَّدة غير مفاجئ . فالقاعدة تولِّد سلاسل بعمليةٍ خوارزميةٍ معرَّفةٍ جيداً (a well-defined algorithmic process) ، وبالتالي فالاشتقاقات يمكن تنفيذها على آلة تيورنج .

ولبيان الاتجاه العكسي نَصِف كيف يمكن لأي آلة تيورنج أن تحاكَى (mimicked) بقاعدة غير مقيَّدة .

نفرض أننا قد أُعطينا آلة تيورنج $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ ، والمطلوب منا أن ننتج قاعدة G بحيث أن $L(G) = L(M)$. والفكرة وراء عملية الإنشاء بسيطة نسبيا ، ولكن تنفيذها يصبح مملا وطويلا من حيث الاصطلاحات .

ونظرا لأن الحسابات في آلة تيورنج يمكن وصفها بمتتابعة الأوصاف اللحظية

$$q_0 w \vdash^* xqfy, \qquad (3)$$

فسنحاول ترتيبها بحيث أن القاعدة المقابلة (corresponding grammar) تكون لها الخاصية (property) أن العلاقة

$$q_0 w \overset{*}{\Rightarrow} xq_f y \qquad (4)$$

تتحقق إذا وفقط إذا تحققت العلاقة (3) . وهذا ليس من الصعب تنفيذه . ولكن ما هو صعب أن نراه هو كيفية إيجاد رابط بين العلاقة (4) وما نريده فعليا ونعني به

$$S \overset{*}{\Rightarrow} w$$

وذلك لجميع السلاسل w التي تحقق العلاقة (3) . وللوصول إلى ذلك ننشئ قاعدة تتحقق فيها الخصائص التالية :

١) S يمكن أن تشتق $q_0 w$ لجميع السلاسل $w \in \Sigma^+$.

٢) العلاقة (4) ممكن تحققها إذا وفقط إذا تحققت العلاقة (3) .

٣) إذا تم توليد سلسلة $xq_f y$ حيث $q_f \in F$ فإن القاعدة تحوِّل (transforms) هذه السلسلة إلى السلسلة الأصلية w .

وعندئذ تكون المتتابعة الكاملة للاشتقاقات (complete sequence of derivations) هي :

$$S \overset{*}{\Rightarrow} q_0 w \overset{*}{\Rightarrow} xq_f y \overset{*}{\Rightarrow} w \qquad (5)$$

الخطوة الثالثة في الاشتقاق أعلاه هي الخطوة التي قد تثير لنا بعض المتاعب أو الصعوبات ، إذ كيف يمكن للقاعدة أن تتذكر w (remember) إذا تم تعديلها (modified) أثناء تنفيذ الخطوة الثانية ؟ نحل هذه المشكلة بتشفير سلاسل (encoding strings) بحيث أن الصيغة المشفَّرة (coded version) تحتوي أصلا (originally) على نسختين من السلسلة w : الأولى يتم حفظها (saved) ، بينما الثانية تُستخدم في الخطوات المشار إليها في العلاقة (4) . وعندما يتم إدخال تشكيلة نهائية (a final configuration) ، فإن القاعدة تمحو كل شئ باستثناء السلسلة w التي تم حفظها .

وكي نقوم بإنتاج (producing) نسختين من السلسلة w ، ونعالج (handle) رمز الحالة (state symbol) في الآلة M [والذي يجب أن يُحذف (removed) في النهاية (eventually) بالقاعدة (grammar)] ، فإننا نُعَرِّف متغيرين V_{ab} و V_{aib} وذلك لجميع العناصر $b \in \Gamma$, $\forall a \in \Sigma \cup \{\square\}$ ، وجميع القيم i بحيث أن $q_i \in Q$. المتغير V_{ab} يُشَفِّر (encodes) الرمزين a, b ، بينما المتغير V_{aib} يُشَفِّر الرمزين a, b وأيضا الحالة q_i .

الخطوة الأولى في العلاقة (5) يمكن تحقيقها [في الصيغة المشفَّرة (encoded form)] بما يلي :

$$S \rightarrow V\square\square S \mid SV\square\square \mid T \qquad (6)$$
$$T \rightarrow TV_{aa} \mid V_{a0a} \qquad (7)$$

لجميع العناصر $\forall a \in \Sigma$. هذه الإنتاجات تسمح للقاعدة بتوليد صيغة مشفرة لأي سلسلة $q_0 w$ بأي عدد اختياري من الفراغات الأمامية والخلفية (leading and trailing blanks) .

وبالنسبة للخطوة الثانية : فلكل انتقال

$$\delta(q_i, c) = (q_j, d, R)$$

للآلة M نضع في إنتاجات القاعدة

$$V_{aic}\ V_{pq} \rightarrow V_{ad}\ V_{pjq} \qquad (8)$$
$$\forall a, p \in \Sigma \cup \{\Box\}, q \in \Gamma.$$

ولكل انتقال

$$\delta(q_i, c) = (q_j, d, L)$$

في الآلة M نجعل القاعدة G تشتمل على الإنتاج

$$V_{pq}\ V_{aic} \rightarrow V_{pjq}\ V_{ad} \qquad (9)$$
$$\forall a, p \in \Sigma \cup \{\Box\}, q \in \Gamma.$$

وفي الخطوة الثانية إذا دخلت الآلة M حالة نهائية ، فعندئذ يجب أن تتخلص القاعدة من كل شئ ما عدا السلسلة w التي يتم حفظها في المؤشرات الأولى من المتغيرات (first indices of the) $V's$. ولذلك فلكل $q_j \in F$ نضع إنتاجات

$$V_{ajb} \rightarrow a \qquad (10)$$
$$\forall a \in \Sigma \cup \{\Box\}, b \in \Gamma.$$

وهذه تُنشئ أول رمز طَرَفِي (terminal) في السلسلة ، والذي يتسبَّب عندئذ في تغيير / إعادة كتابة (rewriting) في بقية السلسلة بواسطة الإنتاجات

$$cV_{ab} \rightarrow ca, \qquad (11)$$
$$V_{ab}c \rightarrow ac, \qquad (12)$$

$$\forall a, c \in \Sigma \cup \{\Box\}, b \in \Gamma.$$

ونحتاج زيادة على ما سبق إلى إنتاج خاص وهو

$$\Box \rightarrow \lambda \qquad (13)$$

وهذا الإنتاج الأخير يأخذ في الاعتبار حالة تحرك الآلة M خارج نطاق جزء الشريط الذي تحتله المدخلات w . وكي تأخذ الأمور مجراها في هذه الحالة يجب أن نستخدم أولا العلاقتين (6) و (7) لنولِّد

$$\square \ldots \square q_0 w \square \ldots \square$$

والتي تمثِّل منطقة الشريط المستخدمة كلها . وأما الفراغات المتفرقة (extraneous blanks) فتُحذف في النهاية بالإنتاج (13) .

والمثال التالي يوضح طريقة الإنشاء المعقَّدة هذه . ويمكن للقارئ الكريم أن يمعن النظر في كل خطوة في هذا المثال ليرى وظيفة الإنتاجات المختلفة ولماذا نحتاج إليها .

مثال ٣ – ١ :

نفرض أن $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ آلة تيورنج حيث

$$\begin{aligned} Q &= \{q_0, q_1\}, \\ \Gamma &= \{a, b, \square\}, \\ \Sigma &= \{a, b\}, \\ F &= \{q_1\}, \end{aligned}$$

وكذلك

$$\begin{aligned} \delta(q_0, a) &= (q_0, a, R), \\ \delta(q_0, \square) &= (q_1, \square, L). \end{aligned}$$

هذه الآلة تقبل اللغة $L(aa^*)$.

الآن نأخذ في الاعتبار العمليات الحسابية

$$q_0aa \vdash aq_0a \vdash aaq_0\square \vdash aq_1a\square \qquad (14)$$

التي تقبل السلسلة aa . ولاشتقاق هذه السلسلة بواسطة القاعدة G (grammar) نستخدم أولا القواعد (rules) التي صيغتها (6) و (7) لنحصل على السلسلة الابتدائية المناسبة (appropriate starting string)

$$S \Rightarrow SV_{\square\square} \Rightarrow TV_{\square\square} \Rightarrow TV_{aa}V_{\square\square} \Rightarrow V_{a0a}V_{aa}V_{\square\square}$$

الصيغة الحارسة الأخيرة (the last sentential form) هي نقطة البداية (starting point) لجزء الاشتقاق الذي يحاكي (mimics) حسابات آلة تيورنج . وهي تحتوي على

المدخلات الأصلية $aa\square$ (original input) في متتابعة المؤشرات الأولى (first indices) والوصف اللحظي الابتدائي $q_0aa\square$ (initial instantaneous description) في المؤشرات المتبقية (remaining indices) . وبعد ذلك نطبق الإنتاج

$$V_{a0a}\, V_{aa} \rightarrow V_{aa}\, V_{a0a}$$

والإنتاج

$$V_{a0a}\, V_{\square\square} \rightarrow V_{aa}\, V_{\square 0\square}$$

وكل من هذين الإنتاجين يُعَدُّ حالة خاصة من الإنتاج (8) ، والإنتاج

$$V_{aa}\, V_{\square 0\square} \rightarrow V_{a1a}\, V_{\square\square}$$

الذي يأتي من (9) . تلي ذلك الخطوات التالية في الاشتقاق

$$V_{a0a}\, V_{aa}\, V_{\square\square} \Rightarrow V_{aa}\, V_{a0a}\, V_{\square\square} \Rightarrow V_{aa}\;\; V_{aa}\, V_{\square 0\square} \Rightarrow V_{aa}\, V_{a1a}\, V_{\square\square}$$

ومتتابعة المؤشرات الأولى (first indices) تبقى هي نفسها ، متذكرةً دائما المدخلات الابتدائية (initial input) . وأما متتابعة المؤشرات الأخرى فهي

$$0aa\square,\ a0a\square,\ aa0\square,\ a1a\square$$

والتي هي مكافئة لمتتابعة الأوصاف اللحظية (instantaneous descriptions) في العمليات الحسابية في (14) .

وأخيرا نستخدم الإنتاجات من (10) إلى (13) في الخطوات الأخيرة

$$V_{aa}\, V_{a1a}\, V_{\square\square} \Rightarrow V_{aa}aV_{\square\square} \Rightarrow V_{aa}a\square \Rightarrow aa\square \Rightarrow aa$$

وعملية الإنشاء (construction) الموصوفة في الإنتاجات من (6) إلى (13) هي أساس برهان النتيجة التالية .

نظرية ٣ – ٧ :

لأي لغة L قابلة للتعديد ارتداديا توجد قاعدة غير مقيَّدة G بحيث أن $L = L(G)$.

البرهان :

عملية الإنشاء التي تم وصفها قبل النظرية تضمن لنا أن

$$X \vdash y$$

والتالي

$$e(x) \Rightarrow e(y)$$

حيث $e(x)$ ترمز إلى تشفير (encoding) سلسلة بناءً على الاصطلاح (convention) المعطَى . وبالاستقراء (induction) على عدد الخطوات يمكننا إثبات أن

$$e(q_0 w) \overset{*}{\Rightarrow} e(y)$$

إذا وفقط إذا كان

$$q_0 w \vdash^{*} y$$

ويجب أيضا أن نثبت أنه يمكننا توليد أي تشكيلة ابتدائية ممكنة (every possible starting configuration) وأن السلسلة w يعاد إنشاؤها بطريقة صحيحة / سليمة (properly reconstructed) إذا وفقط إذا دخلت الآلة M تشكيلة نهائية (a final configuration) . وتفاصيل الإثبات ليست بالغة الصعوبة وسنتركها كتدريب للقارئ الكريم .

* * *

النظريتان السابقتان (نظرية ٣ – ٦ ونظرية ٣ – ٧) تُبَيِّنان أن عائلة اللغات المرتبطة بالقواعد غير المقيَّدة تتطابق مع عائلة اللغات القابلة للتعديد ارتداديا .

ثالثا : اللغات والقواعد الحسَّاسة للسياق

Context - Sensitive Grammars and Languages

بين القواعد المقيَّدة حرة السياق (restricted context-free) ، والقواعد العامة غير المقيَّدة (general unrestricted grammars) هناك قواعد متنوعة كثيرة جدا ونوعاً ما مقيَّدة يمكننا تعريفها . وليست جميع الحالات تؤدي إلى نتائج هامة . ومن تلك التي تؤدي إلى نتائج هامة القواعد الحساسة للسياق التي تولِّد لغات مرتبطة بطبقة مقيَّدة من آلات تيورنج ونعني بها

الآلات المقيَّدة الخطيـة (linear bounded automata) التي عَرَّفناهـا في الفصـل السـابق (تعريف ٢ – ٦) .

تعريف ٣ – ٤ :

يقـال لقاعدة $G = (V, T, S, P)$ إنهـا " قاعدة حسَّاسة للسياق " (context - sensitive grammar) إذا كانت جميع الإنتاجات صيغتها

$$x \to y$$

حيث

$$x, y \in (V \cup T)^+, \quad |x| \le |y| \qquad (15)$$

هذا التعريف يعرض بوضـوح أحد أوجه هذا النوع من القواعد وهو أنه " غير متقلص " / " غـير مـنكمش " (noncontracting) ، بمعنـى أن أطـوال الصـيغ العباراتيـة المتعاقبـة (successive sentential forms) لا يمكن أن تتناقص . وأقل وضوحا من هذا : لماذا يطلق على مثل هذه القواعد أنها " حساسة للسياق " ؟ ، ولكن يمكن إثبات (*) أن هذه القواعد جميعها يمكن إعادة كتابتها بصيغة قياسية (a normal form) تكون فيها جميع الإنتاجات صيغتها

$$x\,A\,y \to xv\,y$$

وهذا يكافئ القول بأن الإنتاج

$$A \to v$$

يمكن أن يطبَّق فقط في الحالة التي تظهر فيها A في سياق تكون فيه السلسلة x على اليسار ، والسلسلة y على اليمين . وبينما نستخدم الاصطلاحات (terminology) الناشئة من هذا التفسير الخاص ، إلا أن الصيغة نفسها ليست ذات أهميـة كبيرة بالنسبة لنا هنا ، وسنعتمد كليَّة على تعريف ٣ – ٤ .

(*) انظر مثلا : A. Salomaa, 1973, Forma1 Languages, New York, Academic Press

اللغات الحسَّـاسة للسياق والآلات المقيَّدة الخطية
Context – Sensitive Languages and Linear Bounded Automata

ترتبط القواعد الحساسة للسياق بعائلة لغات تحمل الاسم نفسه .

تعريف ٣ – ٥ :

يقال لِلُغة L إنها " لغة حسَّاسة للسياق " **(a context-sensitive language)** إذا وُجـــدت قاعدة حسَّاسة للسياق G **(a context -sensitive grammar)** ، بحـيث أن $L = L(G)$ أو $L = L(G) \cup \{\lambda\}$.

في هذا التعريف نعيد تقديم السلسلة الخاوية . فتعريف ٣ – ٤ يقتضي (implies) أن الإنتاج $x \rightarrow \lambda$ غير مسموح به ، وبالتالي فأي قاعدة حسَّاسة للسياق لا يمكنها أبدا توليد لغة تحتوي على السلسلة الخاوية . إلا أن أي لغة حرة السياق بدون السلسلة الخاوية λ يمكننا توليدها بحالة خاصة من قاعدة حسَّاسة للسياق ، ولتكن مثلا بقاعدة في الصيغة القياسية لجرايباخ أو تشومسكي (Chomsky or Greibach normal form) ، وكل منهما تحقق شروط تعريف ٣ – ٤ . فإذا جعلنا السلسلة الخاوية يشملها (included) تعريف اللغة الحسَّاسة للسياق [ولكن ليس القاعدة (grammar)] ، فيمكننا حينئذ ادعاء (claiming) أن عائلة اللغات حرة السياق مجموعة جزئية من عائلة اللغات الحسَّاسة للسياق .

مثال ٣ – ٢ :

اثبت أن اللغة $L = \{a^n b^n c^n : n \geq 1\}$ لغة حسَّاسة للسياق .

الحل :

سنثبت ذلك بإعطاء قاعدة حسَّاسة للسياق تولد هذه اللغة . إحدى هذه القواعد هي

$$\begin{aligned} S &\rightarrow abc|aAbc, \\ Ab &\rightarrow bA, \\ Ac &\rightarrow Bbcc, \\ bB &\rightarrow Bb, \\ aB &\rightarrow aa|aaA. \end{aligned}$$

ويمكننا توضيح عمل هذه القاعدة ببيان اشتقاقٍ للسلسلة $a^3 b^3 c^3$:

$$S \Rightarrow aAbc \Rightarrow abAc \Rightarrow abBbcc$$
$$\Rightarrow aBbbcc \Rightarrow aaAbbcc \Rightarrow aabAbcc$$
$$\Rightarrow aabbAcc \Rightarrow aabbBbccc$$
$$\Rightarrow aabBbbccc \Rightarrow aaBbbbccc$$
$$\Rightarrow aaabbbccc.$$

نلاحظ أن الحل يستخدم المتغيرين A, B كرسولين (messengers) ، حيث يتم أولا إنشاء رمز A على اليسار ، ثم يتحرك A إلى اليمين إلى أول c حيث ينشئ هناك حرفين آخرين b و c . ثم يرسل الرمزَ الرسولَ B إلى الخلف إلى اليسار لينشئ الحرف المقابل a . والعملية شبيهة جدا بالطريقة التي قد تُتبع لبرمجة آلة تيورنج لقبول اللغة L .

* * *

نظراً لأن اللغة المعطاة في المثال السابق (مثال ٣ – ٢) ليست حرة السياق ، فلذلك نرى أن عائلة اللغات حرة السياق مجموعة جزئية صحيحة (a proper subset) من عائلة اللغات الحسَّاسة للسياق . ومثال ٣ – ٢ يبيِّن أيضا أنه ليس من السهل إيجاد قاعدة حسَّاسة للسياق حتى للأمثلة البسيطة نسبيا . وغالبا ما نحصل على الحل بسهولة بالغة بأن نبدأ ببرنامج آلة تيورنج ، ثم نوجد قاعدة مكافئة له . وستوضح بعض الأمثلة أنه كلما كانت اللغة حسَّاسة للسياق فإن آلة تيورنج المقابلة ستكون لها متطلبات سعة يمكن التنبؤ بها (predictable space requirements) . وبصورة خاصة يمكن النظر إلى هذه الآلة على أنها آلة مقيَّدة خطية (a linear bounded automaton) .

نظرية ٣ – ٨ :

لأي لغة L حسَّاسة للسياق لا تشتمل على السلسلة الخاوية λ توجد آلة M مقيَّدة خطيَّة (linear bounded automaton) بحيث أن $L = L(M)$.

البرهان :

إذا كانت اللغة L حسَّاسة للسياق ، فمعنى ذلك أنه توجد قاعدة G حسَّاسة للسياق للغة $L - \lambda$. وسنُبَيِّن بإذن الله أن الاشتقاقات (derivations) في هذه القاعدة يمكن محاكاتها بآلة مقيَّدة خطية . هذه الآلة سيكون لها مساران (two tracks) : أحدهما يحتوي على سلسلة المدخلات w (input string) ، والآخر يحتوي على الصيغ العباراتية (sentential forms) المشتقة باستخدام القاعدة G . ونقطة هامة تُعَدُّ مفتاحا (key point) لهذا البرهان / الحجة (argument) هي أن أي صيغة عباراتية ممكنة لا يمكن أن يكون طولها أكبر من $|w|$. ونقطة أخرى جديرة بالملاحظة هي أن أي آلة مقيَّدة خطية تكون – بالتعريف – غير محدَّدة

(nondeterministic) . وهذا أمر ضروري في البرهان نظرا لأنه يمكننا أن نَدَّعي أن الإنتاج الصحيح يمكننا دائما تخمينه ، وأنه لا يلزمنا تتبُّع (pursuing) أي بدائل غير منتجة (unproductive alternatives) . ولذلك فإن الحسابات (computation) التي وصفناها في نظرية ٣ – ٦ يمكننا إجراؤها بدون استخدام أي حيِّز عدا هذا الذي تشغله أصلا السلسلة w ، أي أنه يمكننا تنفيذها بآلة مقيَّدة خطية .

نظرية ٣ – ٩ :

إذا تم قبول لغةٍ ما L بآلة مقيَّدة خطية M ، فإنه توجد قاعدة حسَّاسة للسياق G تولِّد اللغة L .

البرهان :

طريقة الإنشاء هنا شبيهة بتلك المذكورة في نظرية ٣ – ٧ . وجميع الإنتاجات التي تم توليدها في نظرية ٣ – ٧ غير متقلصة / غير منكمشة (noncontracting) ما عدا الإنتاج

$$\square \rightarrow \lambda$$

ولكن هذا الإنتاج يمكن حذفه . وهو ضروري فقط عندما تتحرك آلة تيورنج خارج حدود (bounds) المدخلات الأصلية ، وهذه ليست هي الحال هنا . والقاعدة التي نحصل عليها بطريقة الإنشاء بدون هذا الإنتاج غير الضروري هي غير متقلصة / غير منكمشة ، وبذلك يكتمل البرهان .

العلاقة بين اللغات الارتدادية واللغات الحساسة للسياق
Relation between Recursive and Context - Sensitive Languages

نظرية ٣ – ٩ تخبرنا أن أي لغة حسَّاسة للسياق ستقبلها آلة تيورنج ما ، وبالتالي فهي قابلة للتعديد ارتداديا . والنظرية التالية (نظرية ٣ – ١٠) نصل إليها بسهولة من هذه النتيجة .

نظرية ٣ – ١٠ :

أي لغة L حسَّاسة للسياق هي لغة ارتدادية (recursive) .

البرهان :

نأخذ في الاعتبار اللغة L الحسَّاسة للسياق مع قاعدة G حسَّاسة للسياق مرتبطة باللغة L ، وننظر إلى اشتقاق للسلسلة w

$$S \Rightarrow x_1 \Rightarrow x_2 \Rightarrow \ldots \Rightarrow x_n \Rightarrow w$$

يمكننا أن نفرض – دون أي فقدان للعمومية (any loss of generality) – أن جميع الصيغ العباراتية (sentential forms) في اشتقاق مفرد (a single derivation) مختلفة ، أي أن $x_i \neq x_j \ \forall \ i \neq j$. والنقطة الأساسية في البرهان هي أن عدد الخطوات في أي اشتقاق هو دالة محدودة في $| w |$ (a bounded function of) . نعلم أن

$$| x_j | \leq | x_j + 1 |$$

لأن القاعدة G غير متقلصة (غير منكمشة) . والشئ الوحيد الذي نحتاج لإضافته هو أنه يوجد عددٌ ما m – يعتمد فقط على القاعدة G والسلسلة w – بحيث أن

$$| x_j | < | x_{j+ m} |$$

لجميع قيم j ، حيث $m = m (| w |)$ دالة محدودة في (a bounded function of) $| w |$ & $| V \cup T |$. وهذا يرجع إلى أن محدودية (finiteness of) $| V \cup T |$ تقتضي (implies) أنه يوجد عدد محدود (a finite number) فقط من السلاسل ذوات أي طول معطَى (strings of a given length) . ولذلك فإن طول أي اشتقاق لسلسلة $w \in L$ هو على الأكثر $| w | \ m (| w |)$.

هذه الملاحظة تعطينا فورا خوارزمية عضوية / انتماء لِلُّغةِ (a membership algorithm for) L : نختبر (check) جميع الاشتقاقات التي لا يزيد طول أي منها عن $| w | \ m (| w |)$. ونظرا لأن مجموعة إنتاجات القاعدة G محدودة (finite) ، فلذلك يوجد عدد محدود فقط من هذه الاشتقاقات . فإذا أَعْطَى أَيٌّ منها السلسلة w ، فإن $w \in L$ ، وإلا فإن w لا تنتمي لهذه اللغة L .

نظرية ٣ – ١١ :

توجد لغة ارتدادية ليست حسَّاسة للسياق .

البرهان :

نأخذ في الاعتبار مجموعة جميع القواعد الحسَّاسة للسياق على $T = \{ a, b \}$. يمكننا استخدام اصطلاح تحتوي فيه كل قاعدة على مجموعة متغيرات صيغتها

$$V = \{ V_0, V_1, V_2, \ldots \}$$

وكل قاعدة حسَّاسة للسياق يتم تحديدها (specified) تماما بإنتاجاتها . ويمكننا أن ننظر إلى هذه الإنتاجات على أنها مكتوبة كسلسلة مفردة (a single string) :

$$x_1 \rightarrow y_1 \; ; x_2 \rightarrow y_2; \ldots; x_m \rightarrow y_m$$

والآن نطبق على هذه السلسلة التشاكل (homomorphism) التالي

$$\begin{aligned} h\,(a) &= 010, \\ h\,(b) &= 01^20, \\ h\,(\rightarrow) &= 01^30, \\ h\,(\;;\;) &= 01^40, \\ h\,(V_i) &= 01^{i+5}0. \end{aligned}$$

وهكذا فأي قاعدة حسَّاسة للسياق يمكن أن تُمثَّل بطريقة وحيدة (uniquely) بسلسلة (string) من $L((011^*0)^*)$. ويضاف إلى ذلك أن هذا التمثيل قابل للعكس (invertible) ، بمعنى أنه إذا أُعطِينا أي سلسلة من هذه السلاسل ، فإنه توجد على الأكثر قاعدة واحدة حسَّاسة للسياق مقابلة لهذه السلسلة .

دعنا الآن نقدم ترتيبا صحيحاً على (a proper ordering on) $\{0, 1\}^+$ ، وبالتالي يمكننا كتابة سلاسل بالترتيب $w_1, w_2, \ldots$. أيُّ سلسلة معطاة w_j قد لا تُعَرِّف قاعدة حساسة للسياق ، فإذا ما عَرَّفت ، فأَطلق على هذه القاعدة G_j . وبعد ذلك نُعَرِّف لغةً L بما يلي :

$$L = \{w_i : w_i \text{ defines a context-sensitive grammar } G_i \text{ and } w_i \notin L\,(G_i)\}$$

أي أن اللغة L تتكون من مجموعة السلاسل التي تُعَرِّف أيٌّ منها قاعدةً حسَّاسةً للسياق ، ولا تنتمي السلسلة إلى اللغة التي تولدها هذه القاعدة . نلاحظ أن اللغة L مُعَرَّفة تعريفا جيدا وهي في الحقيقة ارتدادية . وكي نرى ذلك نقوم بإنشاء خوارزمية عضوية / انتماء : إذا أُعطينا سلسلة w_i ، فإننا نختبرها لنرى إن كانت تُعَرِّف قاعدةً G_i حسَّاسة للسياق . فإن كانت لا تُعَرِّف فإن $w_i \notin L$. وإن كانت السلسلة تُعَرِّف فعلاً قاعدة ، فإن اللغة $L\,(G_i)$ تكون ارتدادية ، ويمكننا استخدام

خوارزمية الانتماء / العضوية المذكورة في برهان نظرية ٣ – ١٠ لنرى ما إذا كانت $w_i \in L(G_i)$ ، أي لنرى ما إذا كانت السلسلة w_i تنتمي إلى اللغة $L(G_i)$. فإن لم تكن تنتمي إلى هذه اللغة ، فإن w_i تنتمي إلى اللغة L .

ولكن اللغة L ليست حسَّاسة للسياق . وذلك لأنها لو كانت حسَّاسة للسياق فستوجد سلسلةٌ ما w_j بحيث أن $L = L(G_j)$. ويمكننا عندئذ أن نسأل ما إذا كانت السلسلة w_j موجودة في اللغة $L(G_j)$. فإذا فرضنا أن $w_j \in L(G_j)$ ، فبناءًا على التعريف نرى أن w_j ليست في اللغة L . ولكن $L = L(G_j)$ ، وبالتالي نصل إلى تناقض (a contradiction) . وبالعكس إذا فرضنا أن $w_j \notin L(G_j)$ ، فبناءًا على التعريف نرى أن $w_j \in L$ ، ونصل إلى تناقض آخر . وهكذا نستنتج حتما أن اللغة L ليست حسَّاسة للسياق .

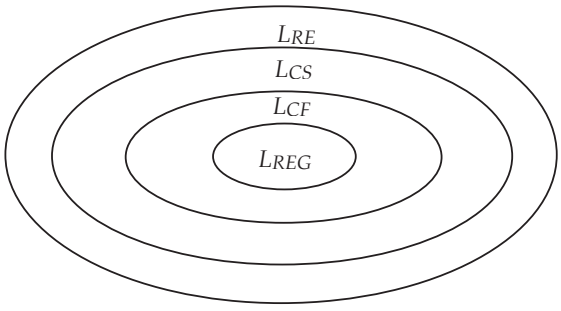
* * *

النتيجة المذكورة في نظرية ٣ – ١١ تشير إلى أن الآلات المقيَّدة الخطية هي فعلا أقل قدرة / قوة من آلات تيورنج ، نظرا لأنها تقبل فقط مجموعة جزئية صحيحة من اللغات الارتدادية . ومن النتيجة نفسها نَخْلُصُ إلى أن الآلات المقيَّدة الخطية هي أقوى وأكثر قدرة من آلات الدفع لأسفل (pushdown automata) . واللغات حرة السياق – نظراً لأنه تُوَلِّدُها قواعد حرة السياق – تُعَدُّ مجموعة جزئية من اللغات الحسَّاسة للسياق . فكما تبيَّن أمثلة عديدة هي مجموعة جزئية صحيحة . وبسبب التكافؤ الأساسي (essential equivalence) بين الآلات المقيَّدة الخطية واللغات الحسَّاسة للسياق من ناحية ، وآلات الدَّفع لأسفل واللغات حرة السياق من ناحية أخرى ، نرى أن أي لغة تقبلها آلة دفع لأسفل ستقبلها أيضا آلة مقيَّدة خطية ما ، ولكن نرى أيضا أنه توجد لغات تقبلها بعض الآلات المقيَّدة الخطية وليس لها أي آلات دفع لأسفل .

رابعا : التدرج الهرمي لتشومسكي The Chomsky Hierarchy

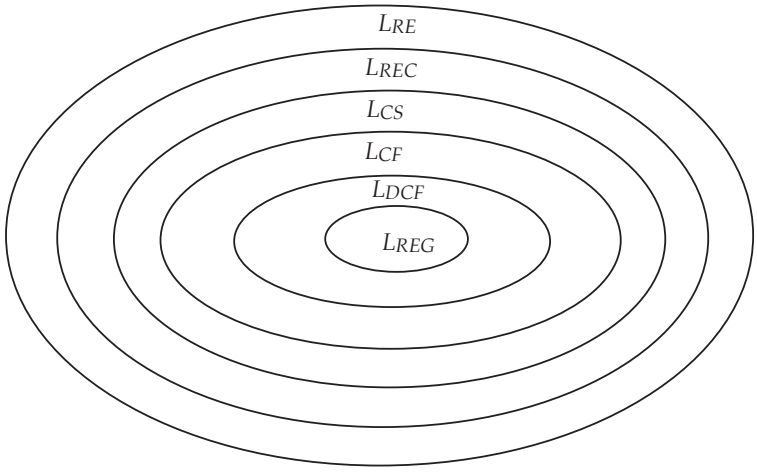
تعرضنا في دراستنا للغات الشكلية لعدد من عائلات اللغات ، من بينها اللغات القابلة للتعديد ارتداديا " L_{RE} " (recursively enumerable languages) ، واللغات الحسَّاسة للسياق " L_{CS} " (context - sensitive languages) ، واللغات حرة السياق "L_{CF}" (context - free languages) ، واللغات المنتظمة " L_{REG} " (regular languages) . وإحدى طرق عرض العلاقة بين هذه العائلات يطلَق عليها " التدرج الهرمي لتشومسكي " **(Chomsky hierarchy)** . وقد وضع " تشومسكي " (Noam Chomsky) – وهو أحد مؤسسي نظرية اللغات الشكلية – تقسيما / تصنيفا مبدئيا (initial classification) لهذه العائلات يتكون من أربعة أنواع لِلُّغات : من النوع

type 0 إلى النـــوع type 3 . وهـــذه الاصطلاحـات الأصلية (original terminology) ظلت متداوَلة أمدا طويلا ، ولا زلنا نجد إشارات متكررة لها ، ولكن الأنواع العددية (numeric types) هي فعليا أسماء أخرى مختلفة لعائلات اللغات التي درسناها . فاللغات نوع 0 هي تلك اللغات التي تولدها قواعد غير مقيَّدة (unrestricted grammars) ، أي اللغات القابلة للتعديد ارتداديا . والنوع 1 يتكون من اللغات الحساسة للسياق ، والنوع 2 يتكون من اللغات حرة السياق ، والنوع 3 يتكون من اللغات المنتظمة . وكما رأينا سابقا فإن أي عائلة لغات من النوع i هي مجموعة جزئية صحيحة من العائلة من النوع $i - 1$. والمخطط التالي (شكل ٣ – ٣) يعرض هذه العلاقة بوضوح ، وهو يبيِّن التدرج الهرمي الأصلي لتشومسكي .



شكل ٣ ـ ٣

وقد تعرضنا سابقا أيضا لعائلات لغات عديدة أخرى يمكن إدراجها في هذا المخطط . وبإدخال عائلة اللغات المحددة حرة السياق " L_{DCF} " (deterministic context-free languages) ، وعائلة اللغات الارتدادية " L_{REC} " نحصل على التدرج الهرمي الموسَّع (extended hierarchy) المبين في مخطط شكل ٣ – ٤ .



شكل ٣ ـ ٤

ويمكن تعريف عائلات لغات أخرى ودراسة موضعها في شكل ٣ – ٤ ، رغم أن علاقاتها لا تتبع بالضرورة دائما البنية المتداخلة (nested structure) البسيطة الواضحة المعالم التي تظهر في شكلي ٣ – ٣ و ٣ – ٤ ، ففي بعض الحالات لا تكون هذه العلاقات مفهومة تماما .

مثال ٣ – ٣ :

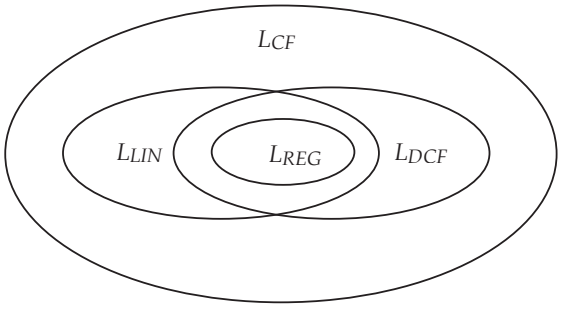
عَرَّفنا سابقا اللغة حرة السياق

$$L = \{w : n_a(w) = n_b(w)\}$$

وأثبتنا أنها محددة (deterministic) ، ولكنها ليست خطية (linear) . ومن ناحية أخرى فإن اللغة

$$L = \{a^n b^n\} \cup \{a^n b^{2n}\}$$

خطية ، ولكنها ليست محددة ، وهذا يشير إلى أن العلاقة بين اللغات المنتظمة ، والخطية ، والمحددة حرة السياق ، وغير المحددة حرة السياق هي كما يوضحها شكل ٣ – ٥ .



شكل ٣ ـ ٥

* * *

تلخيصا لما سبق دراسته لاكتشاف العلاقات بين عدة عائلات من اللغات والآلات المصاحبة لها نقول إن هناك تدرجا / تسلسلا هرميا لِلُّغات والآلات المصنَّفة (classified) بناءً على قوتها / قدرتها (power) كآلات قبول للغة (language accepters) . فآلات تيورنج أقوى / أكثر قدرة من الآلات المقيَّدة الخطية . وهذه بدورها أقوى من آلات الدفع لأسفل . وعند أسفل التدرج الهرمي نجد آلات القبول المحدودة (finite accepters) والتي بَدَأْنا بها دراستنا .

تمرينات رقم (٣)

أولا : اللغات الارتدادية واللغات القابلة للتعديد ارتداديا

٣ – ١) أثبت أن مجموعة جميع اللغات غير القابلة للتعديد ارتداديا ليست قابلة للعد .

٣ – ٢) أثبت أن عائلة اللغات القابلة للتعديد ارتداديا مغلقة (closed) تحت تأثير الاتحاد (union) .

٣ – ٣) اثبت أن مكمِّلة أي لغة حرة السياق يجب أن تكون لغة ارتدادية .

٣ – ٤) إذا كانت L لغة ارتدادية ، فهل صحيح بالضرورة أن اللغة L^+ ارتدادية أيضا ؟

٣ – ٥) لماذا لا يصلح برهان نظرية ٣ – ١ عندما تكون المجموعة S محدودة (finite) ؟

ثانيا : القواعد غير المقيَّدة

٣ – ٦) ما هي اللغة التي تولدها القاعدة غير المقيَّدة التالية ؟

$$
\begin{aligned}
S &\to S_1B,\\
S_1 &\to aS_1b,\\
bB &\to bbbB,\\
aS_1b &\to aa,\\
B &\to \lambda
\end{aligned}
$$

٣ – ٧) نفرض أننا سنقوم بإجراء تعديل على القواعد بحيث أن نقطة البداية (starting point) لأي اشتقاق (derivation) يمكن أن تكون مجموعة محدودة من السلاسل (a finite set of strings) بدلاً من متغير مفرد (a single variable) . قم بصياغة هذا المفهوم شكليا (formalize this concept) ، ثم ادرس العلاقة بين مثل هذه القواعد والقواعد غير المقيَّدة التي استخدمناها في هذا الفصل .

٣ – ٨) اثبت أنه لأي قاعدة غير مقيَّدة توجد قاعدة غير مقيَّدة مكافئة جميع إنتاجاتها صيغتها

$$u \to v$$

حيث

$$u, v \in (V \cup T)^+ \quad \& \quad |u| \le |v|$$

أو

$$A \to \lambda$$

حيث

$$A \in V$$

ثالثا : اللغات والقواعد الحسَّاسة للسياق

٣ – ٩) أوجد قاعدة حسَّاسة للسياق للغة التالية

$$L = \{a^n b^m c^n d^m : n \ge 1, m \ge 1\}$$

٣ – ١٠) اثبت أن عائلة اللغات الحسَّاسة للسياق مغلقة تحت تأثير العكس (closed under reversal).

٣ – ١١) اثبت أنه توجد قاعدة حسَّاسة للسياق للغة

$$L = \{ wuw^R : w, u \in \{a, b\}^+, |w| \ge |u| \}$$

وذلك دون إنشاء (constructing) هذه القاعدة صراحةً (explicitly).

الفصل الرابع

حدود العمليات الحسابية الخوارزمية
Limits of Algorithmic Computation

تكلمنا سابقا عما يمكن لآلات تيورنج أن تقوم بعمله . والآن ننظر إلى ما تعجز هذه الآلات عن أدائه . ورغم أن رسالة تيورنج تدفعنا إلى الاعتقاد بأنه توجد قيود قليلة تحد من قدرة آلة تيورنج ، إلا أننا قد زعمنا في مناسبات عديدة أنه لا يمكن أن توجد أي خوارزميات لحل مسائل معينة . والآن نوضح ما نعنيه بهذا الزعم . فبعض النتائج ظهرت بمنتهى البساطة :إذا كانت لغةٌ ما غير ارتدادية (nonrecursive) فبناءًا على التعريف لا توجد أي خوارزمية انتماء / عضوية (a membership algorithm) لهذه اللغة . وإذا كان هذا هو كل ما هنالك فيما يتعلق بهذا الموضوع فلن تكون لهذا الأمر أهمية تُذكر ، فاللغات غير الارتدادية قيمتها العملية بسيطة . ولكن المسألة أعمق من هذا . فمثلا قد ذكرنا سابقا – ولكن لم نُثبت للآن – أنه لا توجد أي خوارزمية لتحديد ما إذا كانت قاعدة حرة السياق غير مبهمة (unambiguous) أم لا . ومن الواضح أن هذا السؤال له أهمية عملية في دراسة لغات البرمجة .

وسنقوم أولا بتعريف مفهوم " القابلية للحسم / للتقرير / للتحديد " (**decidability**) و " القابلية للحساب " (**computability**) لبيان ما نعنيه حين نقول أن شيئا ما لا يمكن تنفيذه بآلة تيورنج. ثم ننظر بعد هذا إلى عدة مسائل تقليدية من هذا النوع ، ومن بينها مسألة توقف (halting problem) آلات تيورنج والمعلومة جيدا . ويتبع هذه المسألة عدد من المسائل المرتبطة بها وذوات العلاقة لآلات تيورنج واللغات القابلة للتعديد ارتداديا . وبعد ذلك نناقش بعض الأسئلة المتعلقة باللغات حرة السياق . وهنا سنجد عددا من المسائل الهامة والتي لا توجد لها للأسف أي خوارزميات .

أولا : بعض المسائل التي لا يمكن حلها بآلات تيورنج
Some Problems that cannot be solved by Turing Machines

ليس من المفاجئ الادعاء بأن قدرة العمليات الحسابية الميكانيكية محدودة . فبديهيا نعلم أن هناك أسئلة عن أشياء نتوقع حدوثها وغير محددة تماما تتطلب دراسة خاصة ونظرة منطقية فوق قدرة وسعة أي حاسوب يمكننا الآن بناؤه أو حتى توقع بنائه . وما يهم علماء الحاسوب أكثر من

هذا هو أن هناك أسئلة يمكن طرحها بوضوح وببساطة مع إمكانية واضحة لوجود حل خوارزمي (an algorithmic solution) ، ولكن من المعلوم أنه لا يمكن حلها باستخدام أي حاسوب (unsolvable by any computer) .

القابلية للحوسبة والقابلية للحسم

Computability and Decidability

ذكرنا في تعريف ١ – ٤ أنه يقال إن دالة f في مجال معين (a certain domain) قابلة للحوسبة (computable) إذا وُجدت آلة تيورنج تحسب قيمة الدالة f لجميع وسطائها (arguments) في مجالها . ويقال إن الدالة غير قابلة للحوسبة (uncomputable) إذا لم توجد مثل هذه الآلة من آلات تيورنج . وقد توجد آلة تيورنج تستطيع حساب الدالة f على جزء من مجالها (part of its domain) ، ولكننا نطلق على الدالة : " قابلة للحوسبة " (computable) فقط إذا كانت هناك آلة تيورنج تحسب الدالة على مجالها كله . ونرى من هذا أنه عندما نصنّف (classify) أي دالة على أنها قابلة أو غير قابلة للحوسبة فيجب أن نكون واضحين تماما في تحديد مجال الدالة .

واهتمامنا هنا سيكون بإذن الله بالأوضاع المبسطة نوعاً ما ، حيث نتيجة أي حوسبة هي " نعم " (yes) أو " لا " (no) بسيطة . وفي هذه الحالة نقول عن مسألة معينة إنها قابلة للحسم / للتقرير **(decidable)** أو غير قابلة للحسم / للتقرير **(undecidable)** . والمسألة **(problem)** تعني بها مجموعة من العبارات المرتبطة ببعضها البعض (a set of related statements) ، حيث أي منها يجب أن تكون إما صحيحة أو خاطئة (true or false) . فمثلا دعنا نأخذ في الاعتبار العبارة التالية : " لأي قاعدة حرة السياق G ، اللغة $L(G)$ تُعد مبهمة " (For a context - free grammar G, the language $L(G)$ is ambiguous) . بالنسبة لبعض القواعد G هذه العبارة تُعد صحيحة ، وبالنسبة للبعض الآخر تُعد خاطئة ، ولكن من الواضح أنه يجب أن يكون لدينا واحدة منهما أو الأخرى . والمسألة (problem) هي أن نقرر ما إذا كانت العبارة صحيحة لأي قاعدة G معطاة . ومرة أخرى هناك مجال مفهوم ضمنا (underlying domain) ، وهو مجموعة جميع القواعد حرة السياق . ونقول إن مسألةً ما قابلة للحسم / للتقرير (decidable) إذا وُجدت آلة تيورنج تعطِي الإجابة الصحيحة لكل عبارة في مجال المسألة .

عندما نذكر نتائج القابلية أو عدم القابلية للحسم ، يجب أن نعلم دائما ما هو المجال الذي نتكلم عنه ، لأن ذلك قد يؤثر على الاستنتاجات . وبصورة أكثر تحديدا فإن مثالاً واحدا (a single instance) لمسألة يُعد دائما قابلا للحسم ، نظرًا لأن الإجابة (answer) هي إما صحيحة T أو خاطئة F (true of false) . وفي الحالة الأولى فإن آلة تيورنج التي تجيب دائما بالإجابة T (true) ستعطينا الإجابة الصحيحة (correct answer) . بينما في الحالة الثانية فإن آلة تيورنج التي تجيب دائما بالإجابة F (false) تكون هي المناسبة (appropriate) . وهذه الإجابة قد تبدو طريفة (a facetious answer) ، ولكنها تؤكد على نقطة هامة . فحقيقة أننا لا نعلم ما هي الإجابة الصحيحة (correct answer) ليس لها أي أهمية ، وما يهم فعلا هو أنه توجد آلة تيورنج ما تعطينا فعليا الاستجابة الصحيحة (correct response) .

مسألة تَوَقُّف آلة تيورنج The Turing Machine Halting Problem

نبدأ بإذن الله ببعض المسائل ذوات الأهمية التاريخية والتي تعطينا في الوقت نفسه نقطة بداية للتوصل إلى نتائج لاحقة . وأفضل المسائل المعلومة في هذا السياق هي مسألة توقف آلة تيورنج (Turing machine **halting problem**) . ويمكننا تلخيص هذه المسألة ببساطة في السؤال التالي : إذا أُعطينا وصفا (description) لآلة تيورنج M ، وسلسلة مدخلات w ، وبدأنا الآلة في التشكيلة الابتدائية $q_0\ w$ (initial configuration) ، فهل ستقوم الآلة M بتنفيذ عملية حسابية (a computation) تتوقف أخيرا (eventually halts) ؟ وبعرض المسألة بطريقة أكثر اختصارا نسأل : إذا تم تطبيق الآلة M على السلسلة w – أو ببساطة (M, w) – فهل ستتوقف M أم لا ؟ ومجال هذه المسألة نعتبره مجموعة جميع آلات تيورنج وجميع السلاسل w ، أي أننا نبحث عن آلة تيورنج واحدة (a single Turing machine) تحقق الشرط التالي : إذا أعطينا الآلة وصفا لآلة اختيارية M (arbitrary) وسلسلة w ، فإن الآلة ستتنبأ (will predict) بتوقُّف أو عدم توقُّف عملية حسابات (computation) الآلة M عند تطبيقها على السلسلة w .

ونحن لا نستطيع الوصول إلى الإجابة بمحاكاة (simulating) عمل الآلة M على السلسلة w ، مثلاً بتنفيذه على آلة تيورنج شاملة (a universal Turing machine) ، وذلك لأنه لا يوجد حد (limit) أو قيد على طول الحسابات (length of the computation) . وإذا دخلت الآلة M في عروة لا نهائية ، فإننا مهما انتظرنا طويلا لا نستطيع أبدا التأكُّد من أن الآلة M هي فعلا في عروة . وقد تكون هذه ببساطة حالة عملية حسابية طويلة جدا . وما نحتاج إليه هو

خوارزمية تستطيع تحديد الإجابة الصحيحة لأي آلة M وسلسلة w بإجراء بعض التحليلات (analysis) على وصف الآلة والمدخلات . ولكن كما سنرى الآن لا توجد مثل هذه الخوارزمية .

ولمناقشاتنا التالية نرى أنه من المناسب أن تكون لدينا فكرة دقيقة مُحْكَمَة عما نعنيه بمسألة التوقُّف . ولهذا السبب نعطي فيما يلي تعريفا مُحَدَّدا لما ذكرناه سابقا بصورة بسيطة غير محكمة عن مسألة التوقف .

تعريف ٤ – ١ :

نفرض أن w_M سلسلة تصف آلة تيورنج $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$. ونفرض أن w سلسلة في المجموعة الأبجدية (alphabet) للآلة M . سنفترض أن كِلْتَى السلسلتين w_M و w قد شُفِّرت (encoded) كسلسلة من الأصفار 0's والآحاد 1's (كما اقترحنا سابقا في الفصل الثاني – رابعا) . أيُّ حل (solution) لمسألة التوقُّف هو عبارة عن آلة تيورنج H تُحقِّق الشرط التالي (تعمل بالأسلوب التالي) :

لأي سلسلة w_M وسلسلة w تقوم الآلة H بإجراء الحسابات

$$q_0 w_M w \vdash^* x_1 q_y x_2$$

إذا كانت الآلة M تتوقف عند تطبيقها على السلسلة w ، بينما تقوم H بإجراء الحسابات

$$q_0 w_M w \vdash^* y_1 q_n y_2$$

إذا كانت الآلة M لا تتوقف عند تطبيقها على السلسلة w . وهنا q_y, q_n حالتان نهائيتان في الآلة H .

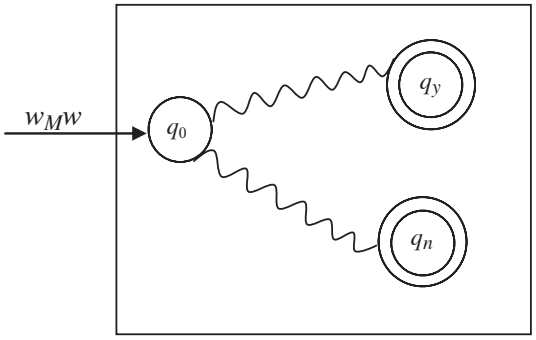
نظرية ٤ – ١ :

لا توجد أي آلة تيورنج H تعمل بالطريقة المطلوبة في تعريف ٤ – ١ . ولذلك فإن مسألة التوقُّف (halting problem) غير قابلة للحسم .

البرهان :

نبرهن النظرية باستخدام مبدأ التناقض . نفرض أنه توجد خوارزمية وبالتالي توجد آلة تيورنج H تحل مسألة التوقف . مدخلات الآلة H ستكون هي السلسلة $w_M\ w$. والمطلوب إذًا هو أنه إذا أُعطينا أي سلسلة $w_M\ w$ ، فإن آلة تيورنج H ستتوقف مع إعطاء إما إجابة : نعم أو إجابة : لا . ونصل إلى هذه النتيجة بطلب أن تتوقف الآلة H في واحدة من حالتين نهائيتين

مقابلتين ، ولتكونا مثلا q_y أو q_n [إشارة إلى q_{yes} أو q_{no}] . ويمكننا أن نصور هذا الوضع بالمخطط المبين في شكل ٤ – ١ . وهدف هذا المخطط هو الإشارة إلى أنه إذا بدأنا الآلة H في الحالة q_0 مع المدخلات $w_M w$ ، فإنها في النهاية ستتوقف إما في الحالة q_y أو في الحالة q_n .



شكل ٤ – ١

وكما هو مطلوب بتعريف ٤ – ١ فإننا نريد من الآلة H أن تعمل بناءً على القاعدة التالية :

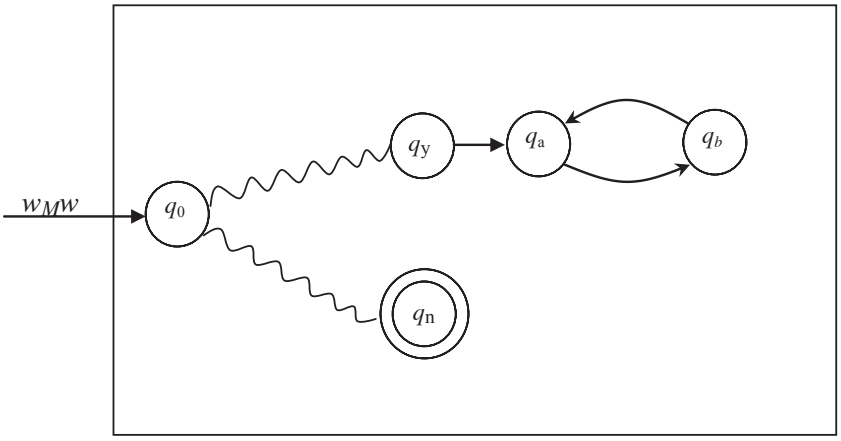
$$q_0 w_M w \vdash^*_H x_1 q_y x_2$$

إذا كانت الآلة M ستتوقف حين تطبيقها على السلسلة w ، بينما

$$q_0 w_M w \vdash^*_H y_1 q_n y_2$$

إذا لم تتوقف M حين تطبيقها على السلسلة w .

وبعد ذلك نقوم بتعديل الآلة H لنحصل على آلة تيورنج H' ذات البنية الموضحة في شكل ٤ – ٢ . وفي هذا الشكل نلاحظ أننا قد أضفنا حالتين q_a, q_b حيث نريد أن تتم الانتقالات (transitions) بين الحالة q_y والحالتين الجديدتين q_a, q_b – بغض النظر عن رمز الشريط (tape symbol) – بطريقة تجعل الشريط يبقى دون تغيير (remains unchanged) . والطريقة التي يتم بها ذلك طريقة مباشرة . فبمقارنة الآلتين H, H' نرى أنه في الحالات التي



شكل ٤ – ٢

تصل فيها الآلة H إلى الحالة q_y وتتوقف ، فإن الآلة المُعَدَّلة H' ستدخل في عروة لا نهائية . وشكليًّا (formally) يوصف عمل الآلة H' كما يلي :

$$q_0 w_M w \vdash^*_{H'} \infty$$

إذا كانت الآلة M ستتوقف حين تطبيقها على السلسلة w ، بينما

$$q_0 w_M w \vdash^*_{H'} y_1 q_n y_2$$

إذا لم تتوقف M حين تطبيقها على السلسلة w .

ومن الآلة H' نُنشئ / نبني (construct) آلة تيورنج أخرى $\hat{H}$. وهذه الآلة الجديدة تأخذ السلسلة w_M كمدخلات وتنسخها ، منتهيةً (ending) في حالتها الابتدائية q_0 . وبعد ذلك تتصرف الآلة $\hat{H}$ تماما مثل الآلة H' . وبعد ذلك يصبح عمل الآلة $\hat{H}$ كما يلي :

$$q_0 w_M \vdash^*_{\hat{H}} q_0 w_M w_M \vdash^*_{\hat{H}} \infty$$

إذا كانت الآلة M ستتوقف حين تطبيقها على السلسلة w_M ، بينما

$$q_0 w_M \vdash^*_{\hat{H}} q_0 w_M w_M \vdash^*_{\hat{H}} y_1 q_n y_2$$

إذا لم تتوقف M حين تطبيقها على السلسلة w_M .

والآن الآلة $\hat{H}$ هي آلة تيورنج ، وبالتالي فلها وَصْف في $\{0, 1\}^*$ (a description in) ، وليكن $\hat{w}$ مثلا . هذه السلسلة – بالإضافة إلى كَوْنِها وَصْفَ الآلة $\hat{H}$ – يمكن استخدامها أيضا كسلسلة إدخال . ولذلك يحق لنا إمكانية التساؤل عما سيحدث إذا ما تم تطبيق الآلة $\hat{H}$ على السلسلة $\hat{w}$. ومما سبق – وبمطابقة الآلة M مع الآلة $\hat{H}$ (identifying M with $\hat{H}$) – نحصل على

$$q_0 \hat{w} \vdash^*_{\hat{H}} \infty$$

إذا كانت الآلة $\hat{H}$ ستتوقف حين تطبيقها على السلسلة $\hat{w}$ ، بينما

$$q_0 \hat{w} \vdash^*_{\hat{H}} y_1 q_n y_2$$

إذا لم تتوقف $\hat{H}$ حين تطبيقها على السلسلة $\hat{w}$. ومن الواضح أن هذا كلام لا معنى له . ومن التناقض الذي وصلنا إليه نستنتج أن فَرْضَنا وجود الآلة H ، وبالتالي فَرْضنا حَسْم مسألة التوقف ، يجب أن يكون خاطئا .

* * *

قد يعترض البعض على تعريف ٤ – ١ نظرا لأننا طلبنا – من أجل حل مسألة التوقف – أنه يجب على الآلة H أن تبدأ وتنتهي في تشكيلةٍ مُعَيَّنة خاصة جدا (very specific configuration) . ولكن ليس من الصعب أن نرى أن هذه الشروط الاختيارية تلعب دورا بسيطا هامشيا في برهان نظرية ٤ – ١ ، وأنه يمكننا استخدام منطق البرهان نفسه مع أي تشكيلات ابتدائية أو نهائية (starting and ending configurations) أخرى . أي أن الاستنتاج الذي وصلنا إليه لا يتأثر بالتعريف الخاص الذي استخدمناه في تعاملنا مع هذه المسألة من أجل تيسير مناقشتها ودراستها .

ومن المهم أن نلاحظ ما تقرره نظرية ٤ – ١ . فهي لا تستبعد حل مسألة التوقف لحالات خاصة معينة ، وفي أحوال كثيرة نستطيع بتحليل كل من الآلة M والسلسلة w أن نخبر ما إذا كانت آلة تيورنج ستتوقف أم لا . وإنما ما تقرره النظرية هو أننا لا نستطيع بصفة دائمة تنفيذ ذلك ، أي أنه لا توجد خوارزمية يمكنها إعطاء قرارٍ فَصْلٍ صحيح لجميع السلاسل w , w_M .

وقد ذَكَرْنا الحجج المعطاة في برهان نظرية ٤ – ١ لأنها تقليدية ولها أهمية تاريخية . وفي الحقيقة فإن النتيجة التي تقررها النظرية تتضمنها فعلا نتائج سابقة كما تبين ذلك الحجة التالية .

نظرية ٤ – ٢ :

إذا كانت مسألة التوقف قابلة للحسم (decidable) ، فكل لغة قابلة للتعديد ارتداديا ستكون لغة ارتدادية . وبالتالي فمسألة التوقف غير قابلة للحسم (undecidable) .

البرهان :

لنرى هذا المذكور في منطوق النظرية ، نفرض أن L لغة قابلة للتعديد ارتداديا على Σ ، ونفرض أن M آلة تيورنج تقبل اللغة L . ونفرض أن H هي آلة تيورنج التي تحل مسألة التوقف . ومن هذا ننشئ الإجراء (procedure) التالي :

(١) طبِّق الآلة H (apply) على السلسلة $w_M\ w$. فإذا قالت الآلة H : " لا " (no) ، فمن التعريف : السلسلة w ليست في اللغة L .

(٢) وإذا قالت الآلة H : " نعم " (yes) ، طبِّق الآلة M على السلسلة w . ولكن الآلة M يجب أن تتوقف ، وبالتالي فستخبرنا في النهاية (eventually) ما إذا كانت السلسلة w في اللغة L أم لا .

وهذا يعطينا خوارزمية عضوية / انتماء (a membership algorithm) جاعلةً اللغة L ارتدادية . ولكننا نعلم يقينا أن هناك لغات قابلة للتعديد ارتداديا وليست ارتدادية . والتناقض الذي وصلنا إليه يقتضي عدم إمكانية وجود الآلة H ، أي أن مسألة التوقف غير قابلة للحسم .

* * *

يلاحظ أن البساطة التي يمكننا بها الحصول على مسألة التوقف من نظرية ٣ – ٥ هي نتيجة تَتْبَع حقيقةَ أنَّ مسألة التوقف ومسألةَ العضوية / الانتماء بالنسبة للغات القابلة للتعديد ارتداديا مسألتان متطابقتان تقريبا . والفارق الوحيد هو أنه في مسألة التوقف لا نُفَرِّق بين التوقف في حالة نهائية وحالة لا نهائية ، بينما في مسألة الانتماء نفرِّق بينهما . وبرهانا نظريتي ٣ – ٥ (عن طريق نظرية ٣ – ٣) و ٤ – ١ قريبا الصلة (closely related) حيث أن كلا منهما يُعَدُّ صيغة من صيغ الإقطار (a version of diagonalization) .

اختزال مسألة غير قابلة للحسم إلى مسألة أخرى
Reducing One Undecidable Problem to Another

ما ذكرناه سلفا عن الارتباط بين مسألة التوقف ومسألة الانتماء يوضح طريقة الاختزال (reduction technique) البالغة الأهمية . حيث نقـــول إن مسألةً ما A تُخْتَزَل إلى مسألة B إذا كانت قابلية A للحَسْم (decidability of A) تَتْبَع (follows from) قابلية B للحسم. وهكذا فإذا عَلِمنا أن A غير قابلة للحسم ، فيمكننا أن نستنتج أن B أيضا غير قابلة للحسم . وفيما يلي ندرس بعض الأمثلة التي توضح هذه الفكرة .

مثال ٤ – ١ : " مسألة دخول الحالة " **(state - entry problem)**

نفـــرض أننـــا قد أُعطينا أي آلة تيورنج $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ ، وأي حالة $q \in Q$ ، وأي سلســـلة $w \in \Sigma^+$. حَـــدِّد / قَرِّر (decide) مـــا إذا كانـــت الحـــالة q سيتـــم دخولهـــا على الإطـــلاق (ever entered) أم لا ، عندما يتم تطبيق الآلة M على السلسلة w (M is applied to).

هذه المسألة غير قابلة للحسم (undecidable) .

لاختزال مسألة التوقف إلى مسألة دخـول الحالة ، نفرض أن لـدينا خوارزمية A تحل مسألة دخول الحالة . يمكننا حينئذ استخدام الخوارزمية لحل مسألة التوقف . مثلا إذا أُعطينا أي آلة M ، وأي سلسلة w ، فإننا نقوم أولا بتعديل (modifying) الآلة M لنحصل على آلة $\hat{M}$ بطريقة تجعل $\hat{M}$ تتوقف في الحالة q إذا وفقط إذا توقفت الآلة M . ويمكننا تنفيذ ذلك ببساطة بالنظر إلى دالة الانتقال δ (transition function) في الآلة M . فإذا توقفت الآلة M ، فإنها تفعل ذلك لأن بعض الانتقالات $\delta(q_i, a)$ غير مُعَرَّفة (undefined) . وللحصول على الآلة $\hat{M}$ ، فإننا نُغيِّر كل هذه الانتقالات غير المُعَرَّفة إلى

$$\delta(q_i, a) = (q, a, R)$$

حيث q : حالة نهائية . ونطبق خوارزمية دخـول الحالة A على ($\hat{M}$, q, w) . فإذا أجابت الخوارزمية A بنعم (yes) ، أي أنه تم دخول الحالة q ، فإن (M, w) تتوقف . وأما إذا قالت الخوارزمية : لا (no) فإن (M, w) لا تتوقف .

وهكذا فإن الفرض أن مسألة دخول الحالة قابلة للحسم يعطينا خوارزمية لمسألة التوقف . ونظرا لأن مسألة التوقف غير قابلة للحسم ، فإن مسألة دخول الحالة يجب أن تكون هي أيضا غير قابلة للحسم .

مثال ٤ – ٢ : " مسألة التوقف على شريط فارغ " (blank - tape halting problem)

مسألة التوقف على شريط فارغ هي مسألة أخرى يمكن أن تُختزل إليها مسألة التوقف (halting problem) . نفرض أننا قد أُعطِينا آلة تيورنج M . حَدِّد / قَرِّر ما إذا كانت الآلة M ستتوقف أم لا إذا ما بدأناها بشريط فارغ (started with a blank tape) .

هذه المسألة غير قابلة للحسم

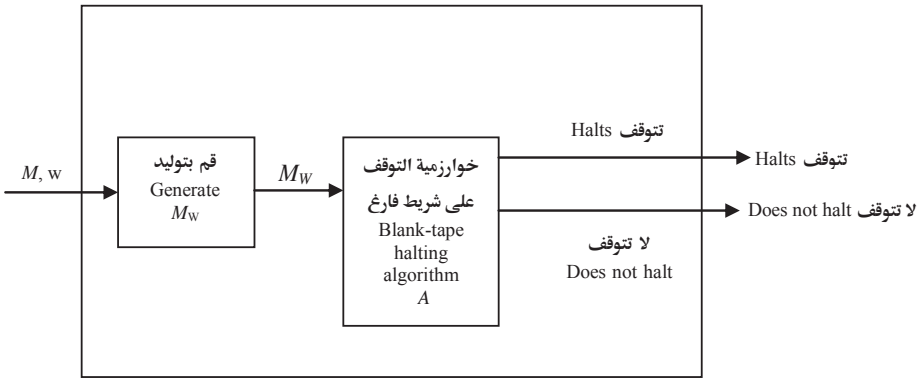
ولبيان كيف يتم هذا الاختزال ، نفرض أننا قد أُعطينا آلة ما M ، وسلسلة ما w . ننشئ أولا من الآلة M آلة جديدة M_w تبدأ بشريط فارغ ، وتكتب عليه السلسلة w ، ثم تضع نفسها (positions itself) في تشكيلة (a configuration) $q_0\, w$. وبعد ذلك تعمل الآلة M_w مثل الآلة M . ومن الواضح أن M_w ستتوقف على شريط فارغ (will halt on a blank tape) إذا وفقط إذا توقفت M على w .

نفـــرض الآن أن مســـألة التوقف على شريط فارغ مسألة قابلة للحسم . إذا أُعطينا أي (M, w) ، فإننا ننشئ أولا M_w ، ثم نطبق عليها خوارزمية مسألة التوقف على شريط فارغ . الاستنتاج (conclusion) النهائي سيخبرنا ما إذا كانت الآلة M حين تُطبق على السلسلة w ستتوقف أم لا . ونظرا لأن هذا يمكن أن يُنفَّذ لأي آلة M وسلسلة w ، فإن أي خوارزمية لمسألة التوقف على شريط فارغ يمكن أن تُحَوَّل (converted) إلى خوارزمية لمسألة التوقف . وحيث أن هذه المسألة الأخيرة معلوم أنها غير قابلة للحسم ، فالنتيجة نفسها يجب أن تكون صحيحة بالنسبة لمسألة التوقف على شريط فارغ .

* * *

عملية البناء / الإنشاء (construction) في برهاني هذين المثالين (مثال ٤ – ١ ومثال ٤ – ٢) توضح طريقة / أسلوبا / منهجا عاما (a common approach) في إثبات / تقرير نتائج عدم القابلية للحسم (establishing undecidability results) . وغالبا ما تساعدنا المخططات القالبية (block diagrams) في توضيح رؤية هذه العملية . فالمخطط المبين في شكل ٤ – ٣ يلخص لنا عملية الإنشاء المذكورة في المثال الأخير (مثال ٤ – ٢) . وفي هذا المخطط نستخدم أولا خوارزمية تنقل (M, w) (transforms) إلى M_w (into) . ومن الواضح أنه

توجد مثل هذه الخوارزمية . وبعد ذلك نستخدم خوارزمية حل مسألة التوقف على شريط فارغ ، ولتكن هي الخوارزمية A مثلا ، والتي نفترض وجودها . وبوضع الاثنتين معا نحصل على خوارزمية لمسألة التوقف . ولكن هذا مستحيل ، وبذلك يمكننا أن نستنتج أن الخوارزمية A لا يمكن أن تكون موجودة .



شكل ٤ – ٣

خوارزمية مسألة التوقف

Algorithm for halting problem

وأي مسألة حسم (a decision problem) هي فعليا دالة مداها {0, 1} (range) ، أي إجابة : صحيح (true) أو خاطئ (false) . ويمكننا أن ننظر أيضا إلى دوال أكثر عمومية لنرى إن كانت هذه الدوال قابلة للحوسبة (computable) أم لا . ولتحقيق ذلك نتبع طريقة الاختزال سالفة الذكر ، ونختزل مسألة التوقف (أو أي مسألة أخرى معلوم أنها غير قابلة للحسم) إلى مسألة حوسبة الدالة قيد الدراسة . وبسبب رسالة تيورنج (Turing's thesis) نتوقع أن تكون الدوال التي نقابلها في الأحوال العملية قابلة للحوسبة . وبالتالي فبالنسبة لأمثلة الدوال غير القابلة للحوسبة علينا أن ننظر في دراستها نظرة أعمق . ومعظم أمثلة الدوال غير القابلة للحوسبة مرتبطة بمحاولات التنبؤ بسلوك آلات تيورنج .

مثال ٤ – ٣ :

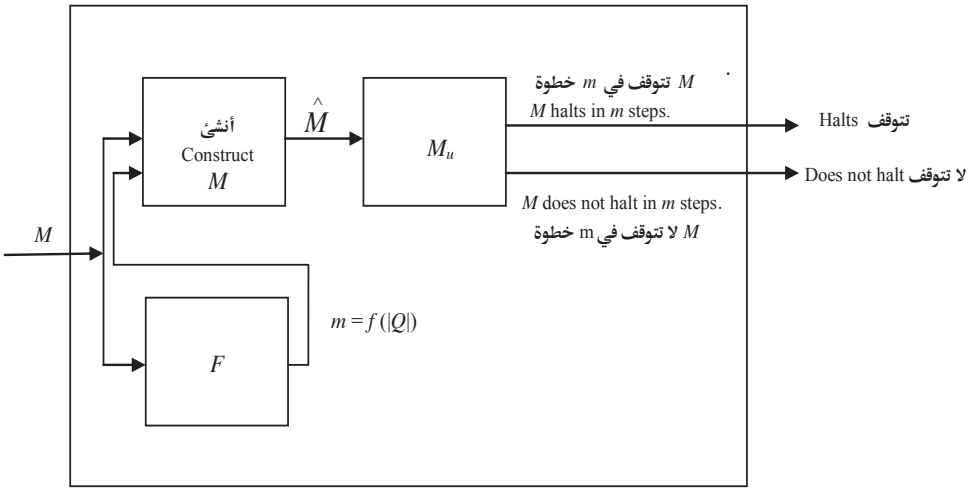
نفرض أن $\Gamma = \{0, 1, \square\}$. ونفرض أن $f(n)$ دالة قيمتها هي أكبر عدد من الحركات (maximum number of moves) التي يمكن أن تقوم بها أي آلة تيورنج ذات n حالة ، وتتوقف حين نبدأها بشريط فارغ .

هذه الدالة – كما سنرى – ليست قابلة للحوسبة .

وقبل أن نشرع في إثبات ذلك ، دعنا نتأكد أن $f(n)$ مُعَرَّفة لجميع قيم n . ولاحظ أولا أنه يوجد عدد محدود فقط من آلات تيورنج التي عدد الحالات في أي منها يساوي n . ويرجع هذا إلى أن كلا من Γ ,Q محدودة (finite) ، وبالتالي فدالة الانتقال δ لها مجال ومدى محدودان (finite domain and range) . وهذا بدوره يقتضي وجود عدد محدود فقط من دوال الانتقال δs المختلفة ، وبالتالي عدد محدود من آلات تيورنج مختلفة ذوات n حالة (different n - state Turing machines) .

ومن بين جميع الآلات ذوات الـ n حالة ، هناك بعض الآلات التي تتوقف دائما (always halt) ، مثلا : الآلات التي لديها حالات نهائية (final states) فقط ، وبالتالي لا تقوم بعمل أي حركات (moves) . وهناك بعض الآلات ذوات الـ n حالة التي سوف لا تتوقف عندما نبدأها بشريط فارغ . ولكنها لا تَدْخل تعريف الدالة f . وأي آلة تتوقف فعلا ستقوم بتنفيذ عدد معين من الحركات ، ومن بين هذه الآلات نأخذ أكبرها لتعطينا $f(n)$.

خذ أي آلة تيورنج M وأي عدد موجب m . من السهل أن نُعَدِّل M (modify) لنحصل على الآلة $\hat{M}$ بطريقة تجعل هذه الأخيرة تتوقف دائما بإحدى إجابتين : إما أن M بتطبيقها على شريط فارغ تتوقف بعد عدد لا يزيد عن m من الحركات (moves) ، أو أن M بتطبيقها على شريط فارغ ستعمل أكثر من m حركة . وكل ما علينا أن نعمله لهذا هو أن نجعل الآلة M تَعُدُّ (count) حركاتها وتتوقف (terminate) عندما يتجاوز هذا العدد القيمة m . افرض



شكل ٤ – ٤

خوارزمية مسألة التوقف على شريط فارغ

Algorithm for blank-tape halting problem

الآن أن $f(n)$ قابلة للحوسبة (computable) بآلة تيورنج ما F . يمكننا بعد ذلك وضع $\hat{M}$ و F معًا كما هو مبيَّن في شكل ٤ – ٤ . نحسب أولا $f(|Q|)$ حيث Q هي مجموعة حالات الآلة M . وهذه القيمة المحسوبة تخبرنا عن أكبر عدد من الحركات (moves) التي يمكن للآلة M أن تعملها إذا كانت ستتوقف . وهذه القيمة التي نحصل عليها تُستخدم بعد ذلك باعتبارها m لإنشاء الآلة $\hat{M}$ كما أشرنا سابقا ، ونعطِي وَصْفًا للآلة $\hat{M}$ لآلة تيورنج شاملة (a universal Turing machine) للتنفيذ . وهذه تخبرنا ما إذا كانت الآلة M حين تطبيقها على شريط فارغ ستتوقف أم لا في أقل من $f(|Q|)$ خطوة . وإذا ما وجدنا أن الآلة M حين تطبيقها على شريط فارغ تعمل أكثر من $f(|Q|)$ حركة ، فإن هذا يقتضي – بسبب تعريف f – أن الآلة M لن تتوقف . وهكذا أصبح لدينا حل لمسألة التوقف على شريط فارغ . واستحالة الاستنتاج (impossibility of the conclusion) تدفعنا إلى القبول بأن الدالة f ليست قابلة للحوسبة .

ثانيا : مسائل غير قابلة للحسم بالنسبة للغات القابلة للتعديد ارتداديا
Undecidable problems for Recursively Enumerable Languages

قررنا سابقا أنه لا توجد خوارزمية انتماء / عضوية بالنسبة للغات القابلة للتعديد ارتداديا . وعدم وجود خوارزمية لتقرير / لحسم خاصيةٍ ما ليس حالة استثنائية بالنسبة للغات القابلة للتعديد ارتداديا ، ولكنه القاعدة العامة . وكما سنرى فيما يلي هناك القليل الذي يمكننا أن نقوله بالنسبة لهذه اللغات . فاللغات القابلة للتعديد ارتداديا عامة جدا بحيث أنه في الأساس أي سؤال نسأله عنها غير قابل للحسم . وعموما حين نسأل سؤالا عن هذه اللغات فإننا نجد أن هناك طريقة ما لاختزال مسألة التوقف إلى هذا السؤال . وفيما يلي نعطي بعض الأمثلة التي توضح كيفية تنفيذ ذلك ، ومن هذه الأمثلة نستنتج إشارة إلى الحالة العامة .

نظرية ٤ – ٣ :

إذا كانت G قاعدة غير مقيَّدة (an unrestricted grammar) ، فإن مسألة تحديد ما إذا كانت

$$L(G) = \varnothing$$

أم لا ، مسألة غير قابلة للحسم .

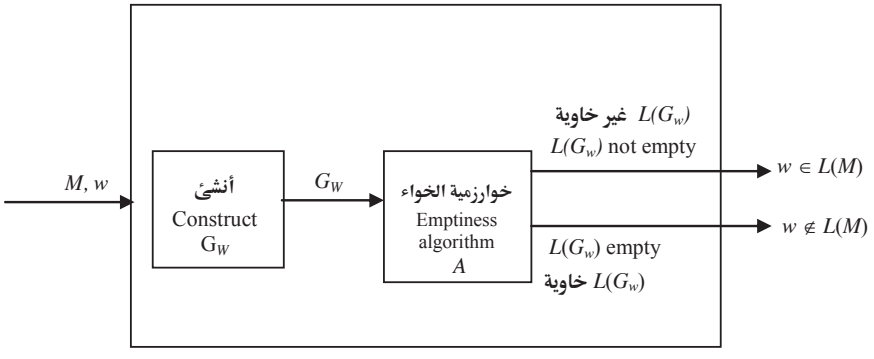
البرهان :

سنختزل مسألة الانتماء / العضوية (membership problem) بالنسبة للغات القابلة للتعديد ارتداديا إلى هذه المسألة . نفرض أننا قد أُعطينا آلة تيورنج M وسلسلة ما w . يمكننا تعديل الآلة M كما يلي . تقوم الآلة M أولا بتخزين مدخلاتها على جزء خاص من شريطها . وبعد ذلك ، كلما دخلت M حالة نهائية ، فإنها تقوم باختبار / بالتحقق (checking) من مدخلاتها المخزونة (saved input) ، وقبولها (accepting it) إذا وفقط إذا كانت هي السلسلة w . ويمكننا تنفيذ ذلك بتغيير دالة الانتقال δ بطريقة بسيطة ، عن طريق إنشاء (creating) آلة M_w لكل سلسلة w بحيث أن

$$L(M_w) = L(M) \cap \{w\}$$

وباستخدام نظرية ٣ – ٧ ننشئ عندئذ قاعدة مقابلة G_w (a corresponding grammar) . ومن الواضح أن عملية الإنشاء التي تقود من الآلة M والسلسلة w إلى القاعدة G_w يمكننا دائما إنجازها . ومن الواضح كذلك أن اللغة $L(G_w)$ تكون غير خاوية (nonempty) إذا وفقط إذا كانت $w \in L(M)$.

والآن نفرض أنه توجد خوارزمية A لتقرير ما إذا كانت $L(G) = \varnothing$ أم لا . إذا فرضنا أن T ترمز إلى خوارزمية نولِّد بها القاعدة G_w ، فإنه يمكننا وضع الخوارزمية T والخوارزمية A معًا كما هو مبين في شكل ٤ – ٥ . وهذا الشكل عبارة عن آلة تيورنج تخبرنا – لأي آلة M وأي سلسلة w – ما إذا كانت السلسلة w تنتمي إلى اللغة $L(M)$ أم لا . وإذا وُجدت مثل آلة تيورنج هذه ، فسيكون لدينا خوارزمية انتماء / عضوية لأي لغة قابلة للتعديد ارتداديا ، وهذا يناقض مباشرة نتيجة أثبتناها سابقا . ولذلك نستنتج أن المسألة المذكورة "$L(G) = \varnothing$" غير قابلة للحسم .



شكل ٤ – ٥

خوارزمية انتماء / عضوية

Membership algorithm

نظرية ٤ – ٤ :

إذا كانت M هي أي آلة تيورنج ، فإن السؤال عما إذا كانت اللغة $L(M)$ محدودة (finite) أم لا غير قابل للحسم .

البرهان :

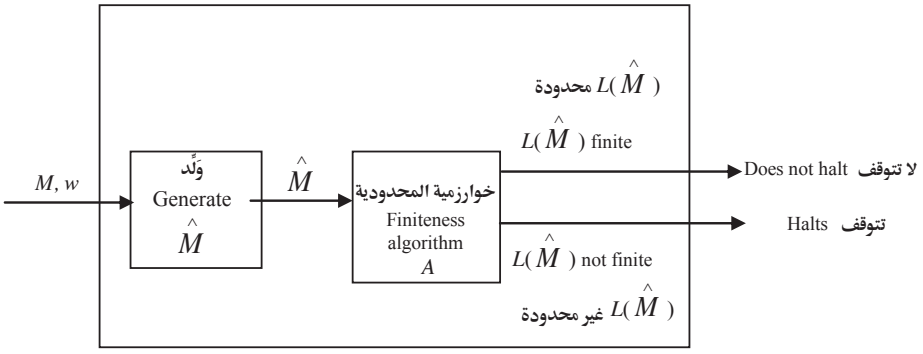
نأخذ في الاعتبار مسألة التوقف (M, w) . من الآلة M ننشئ (construct) آلة تيورنج أخرى $\hat{M}$ تقوم بما يلي :

أولا : حالات التوقف في الآلة M (the halting states of) يتم تغييرها بحيث أنه إذا وَصَلْنا (reached) أيًّا منها ، فإن جميع المدخلات يتم قبولها بالآلة $\hat{M}$. ويمكننا تنفيذ ذلك بجعل أي تشكيلةِ تَوَقُّف (any halting configuration) تذهب إلى حالةٍ نهائية .

ثانيا : يتم تعديل (modifying) الآلة الأصلية بحيث أن الآلة الجديدة $\hat{M}$ تُوَلِّد (generates) أولا السلسلة w على شريطها ، ثم تقوم بإجراء (performing) الحسابات نفسها مثل الآلة M باستخدام السلسلة w التي تم إنشاؤها حديثا (newly created w) ، وبعض الفراغ غير المستخدم عدا ذلك (some otherwise unused space) . وبأسلوب آخر فإن التحركات (moves) التي تقوم بها الآلة $\hat{M}$ بعد أن تكون قد كتبت السلسلة w على شريطها هي نفسها التي كانت ستقوم بها الآلة M لو أنها بدأت في التشكيلة الأصلية $q_0\ w$. وإذا توقفت M في أي تشكيلة ، فإن $\hat{M}$ ستتوقف في حالة نهائية .

ولذلك فإذا توقفت (M, w) ، فإن $\hat{M}$ ستصل إلى حالة نهائية لجميع المدخلات . وإذا لم تتوقف (M, w) ، فإن $\hat{M}$ سوف لا تتوقف أيضا ، وبالتالي سوف لا تقبل (accept) أي شئ . وبأسلوب آخر ، فإن الآلة M ستقبل إما اللغة اللانهائية Σ^+ ، أو اللغة المحدودة $\varnothing$.

فإذا فرضنا الآن وجود خوارزمية A تخبرنا ما إذا كانت اللغة $L(\hat{M})$ محدودة أم لا ، فإنه يمكننا إنشاء (constructing) حل لمسألة التوقف كما هو مُبَيَّن في شكل ٤ – ٦ . وبناءً عليه لا يمكن أن توجد خوارزمية لحسم / لتحديد ما إذا كانت اللغة $L(M)$ محدودة أم لا .



شكل ٤ – ٦

لاحظ في برهان نظرية ٤ – ٤ أن الطبيعة النوعية للسؤال الذي طرحناه – وهو : " هل اللغة $L(M)$ محـــدودة ؟ " – لا أهمية له . ويمكننا تغيير طبيعة المسألة دون التأثير على البرهان بدرجة تُذكر .

مثال ٤ – ٤ :

اثبت أنه لأي آلة تيورنج اختيارية M ذات المجموعة الأبجدية $\Sigma = \{a, b\}$ ، المسألة " اللغة $L(M)$ تحتوي على سلسلتين مختلفتين لهما الطول نفسه " غير قابلة للحسم .

الحل :

نستخدم هنا الطريقة نفسها بالضبط التي اتبعناها في نظرية ٤ – ٤ ، ما عدا أنه عندما تصل الآلة $\hat{M}$ إلى تشكيـــلة تَوَقُّف (a halting configuration) فإنها ستُعَدَّل لتقبل السلسلتين a, b . ولذلك فإن المدخلات الابتدائية (initial input) يتم حفظها (saved) ، وفي نهاية العمليات الحسابية (computation) تقارَن مع a, b ، بحيث يتم قبول هاتين السلسلتين فقط . وهكذا فإذا توقفت (M, w) فإن $\hat{M}$ ستقبل سلسلتين متساويتي الطول ، وما عدا ذلك فإن $\hat{M}$ سوف لا تقبل أي شئ . وأما بقية إثبات المطلوب فتمضي كما في برهان نظرية ٤ – ٤ .

* * *

وبالطريقة نفسها بالضبط يمكننا التعويض (substitution) بأسئلة أخرى مثل : " هل اللغة $L(M)$ تحتوي على أي سلسلة طولها خمسة ؟ " أو " هل اللغة $L(M)$ منتظمة ؟ " دون التأثير بصورة جوهرية على البرهان . هذه الأسئلة ومثيلاتها غير قابلة للحسم . وهناك نتيجة عامة – يطلق عليها نظرية " رايس " (Rice's theorem) – تصيغ (formalizes) ذلك شكليا . وهذه النظرية تنص على أن " أي خاصية غير بسيطة (غير تافهة) (any nontrivial property) من خواص أي لغة قابلة للتعديد ارتداديا غير قابلة للحسم " . وأما الصفة " غير بسيطة أو غير تافهة " فتشير إلى خاصية تتمتع بها بعض اللغات القابلة للتعديد ارتداديا ولكن ليس كل هذه اللغات .

ثالثا : مسألة " بوست " للمقابَلة

The Post Correspondence Problem (PCP)

عـدم القابلية للحسم بالنسبة لمسألة التوقف لـه توابع ونتائج (consequences) عديدة لها أهمية عملية ، وخاصة في مجال اللغات حرة السياق . ولكن في حالات كثيرة يكون من الصعب والممل العمل بمسألة التوقف مباشرة ، ومن الأيسر والأنسب الوصول إلى بعض النتائج الوسطية (intermediate results) التي تسد الفجوة بين مسألة التوقف ومسائل أخرى . وهذه النتائج الوسطية تنتج من عدم قابلية مسألة التوقف للحسم ، ولكنها تُعَدُّ أكثر ارتباطا وقربا من المسائل التي نريد دراستها ، ولذلك فهي تجعل البراهين والإثباتات أيسر وأسهل . وإحدى هذه النتائج الوسطية هي مسألة " بوست " للمقابَلة / للمماثلة (PCP) (the Post Correspondence Problem) .

ومسألة " بوست " للمقابلة (PCP) يمكن عرضها كما يلي :

نفرض أن لدينا متتابعتين (two sequences) تتكون كل منهما من عدد n من السلاسل (strings) على مجموعةٍ أبجديةٍ ما Σ ، ونفرض أن المتتابعتين هما

$$A = w_1, w_2, \ldots w_n$$

$$B = v_1, v_2, \ldots, v_n$$

نقول إنه يوجد حل لمسألة " بوست " للمقابلة (a Post Correspondence solution) – أو اختصارا : حل PC - (PC – solution) – للزوج (A, B) إذا وُجدت متتابعة غير خاوية من أعداد صحيحة $i, j, \ldots, k$ (a nonempty sequence of integers) ، بحيث أن

$$w_i w_j \ldots w_k = v_i v_j \ldots v_k$$

ومسألة PCP هي أن نوجد / نبتكر خوارزمية تخبرنا – لأي زوج (A, B) – ما إذا كان هناك حل - PC أم لا .

مثال ٤ – ٥ – أ)

نفرض أن $\Sigma = \{0, 1\}$ ، وأن

$$A = w_1, w_2, w_3$$

$$B = v_1, v_2, v_3$$

حيث

$$w_1 = 11, w_2 = 100, w_3 = 111,$$
$$v_1 = 111, v_2 = 001, v_3 = 11.$$

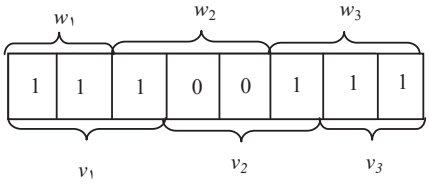
هل للزوج (A, B) حل - PC ؟

ب) أعد حل الجزء أ) بفرض أن

$$w_1 = 00, w_2 = 001, w_3 = 1000,$$
$$v_1 = 0, v_2 = 11, v_3 = 011.$$

الحل :

أ) في هذه الحالة يوجد حل - PC ، كما يبيِّن ذلك شكل ٤ – ٧ .



شكل ٤ – ٧

ب) في هذه الحالة لا يمكن أن يوجد أي حل - PC ، وذلك لأن أي سلسلة مكوَّنة من عناصر من A ستكون أطول من السلسلة المقابلة من B .

* * *

في حالات خاصة قد نقدر على إثبات أن زوجاً ما (A, B) يسمح بوجـود حـل - PC عن طريق إعطاء إنشاء صريح (explicit construction) ، أو قد نقدر على إثبات أنه لا يمكن أن يوجد مثل هذا الحل ، كما فعلنا سابقا (في حل مثال ٤ – ٥ – ب) . ولكن عموما لا توجد خوارزمية لحسم هذا السؤال في جميع الأحوال . ولذلك فإن مسألة PCP غير قابلة للحسم .

وإثبـات ذلك يُعَـدُّ عمليـة طويـلة نوعـاً ما . ومـن أجـل الوضـوح سنقسمهـا إلى جـزئين : في الجـزء الأول نُعـَرِّف مـسألة " بوسـت " المُعَدَّلـة للمقابَلـة **(Modified Post Correspondence Problem) (MPCP)** . ونقـول إن الـزوج (A, B) لـه حـل لمـسألة " بوسـت " المُعَدَّلـة للمقابَلـة (حـل – MPC) (a modified Post correspondence solution) (MPC – solution) إذا وُجدت متتابعة من أعداد صحيحة $i, j, \ldots, k$ ، بحيث أن

$$w_1 w_i w_j, \ldots w_k = v_1 v_i\, v_j \ldots v_k$$

في مسألة " بوست " المُعَدَّلة للمقابَلة (MPCP) العنصران الأوَّلان في المتتابعتين A, B يلعبان دورا خاصا . فأي حل – MPC يجب أن يبدأ بالسلسلة w_1 على الطرف الأيسر ، والسلسلة v_1 على الطرف الأيمن . ولاحظ أنه إذا وُجد حل – MPC فهناك أيضا حل – PC ، ولكن العكس غير صحيح .

ومسألة MPCP هي أن نوجِد / نبتكر خوارزمية لحسم ما إذا كان أي زوج اختياري (A, B) يسمح بوجود حل – MPC أم لا . وهذه المسألة أيضا غير قابلة للحسم . وسوف نثبت عدم قابلية مسألة MPCP للحسم بأن نختزل (reduce) إليها مسألة معلوم عدم قابليتها للحسم ، وهـي مسألة الانتمـاء / العضـوية (membership problem) بالنسبة للغات القابلة للتعديد ارتداديا . ولتحقيق ذلك نُعَرِّف طريقة الإنشاء (construction) التالية . نفرض أننا قد أُعطينا قاعـدة غـير مُقيَّـدة $G = (V, T, S, P)$ (unrestricted grammar) ، وسـلسلة هدفاً w (a target string) . نقوم بإنشاء الزوج A, B كما هو مبيَّن في الشكل التالي (شكل ٤ – ٨) . وفي هذا الشكل نأخذ السلسلة $FS \Rightarrow$ على أنها w_1 ، والسلسلة F على أنها v_1 . وأما ترتيب (order) بقية السلاسل فغير مهم .

A	B	
$FS \Rightarrow$	F	F رمز غير موجود في VUT
a	a	$\forall\ a \in T$
V_i	V_i	$\forall\ V_i \in V$
E	$\Rightarrow wE$	E رمز غير موجود في VUT
y_i	x_i	$\forall\ x_i \rightarrow y_i$ in P
$\Rightarrow$	$\Rightarrow$	

شكل ٤ – ٨

ونريد أن نزعم في النهاية أن $w \in L(G)$ إذا وفقط إذا كان للمجموعتين A, B المُنْشَأَتَيْن (constructed) بهذه الكيفية حل - MPC . ونظرا لأن هذا قد لا يكون واضحا مباشرة فسنوضحه بإذن الله بمثال بسيط .

مثال ٤ – ٦ :

نفرض أن $G = (\{A, B, C\}\ ,\ \{a, b, c,\}\ ,\ S, P)$ ، حيث الإنتاجات P هي

$$S \rightarrow aABb|Bbb,$$
$$Bb \rightarrow C,$$
$$AC \rightarrow aac.$$

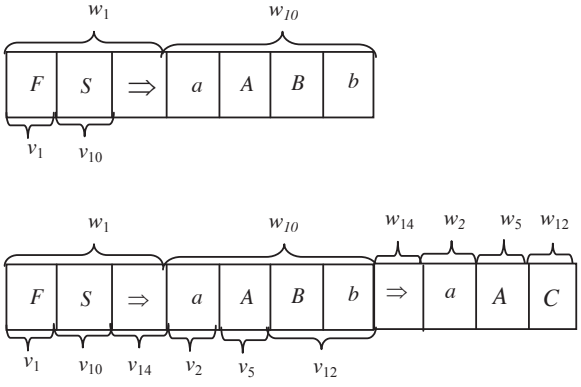
ونأخذ $w = aaac$. المتتابعتان A, B (sequences) اللتان نحصل عليهما من طريقة الإنشاء المقترحة (suggested construction) مبينتان في شكل ٤ – ٩ . السلسلة $w = aaac$ موجودة في اللغة $L\ (G)$ ، ولها اشتقاق

$$S \Rightarrow aABb \Rightarrow aAC \Rightarrow aaac$$

i	w_i	v_i
1	$FS \Rightarrow$	F
2	a	a
3	b	b
4	c	c
5	A	A
6	B	B
7	C	C
8	S	S
9	E	$\Rightarrow aaacE$
10	$aABb$	S
11	Bbb	S
12	C	Bb
13	aac	AC
14	$\Rightarrow$	$\Rightarrow$

شكل ٤ – ٩

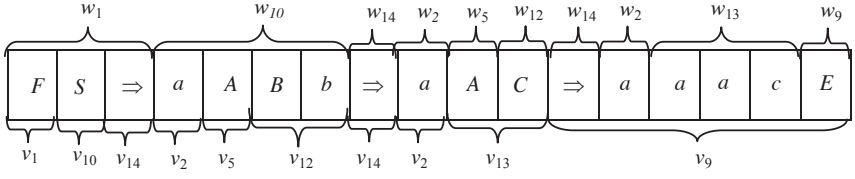
وأما كيفية موازاة (paralleling) هذا الاشتقاق بحل - MPC بواسطة المجموعات المُنْشَأة (constructed sets) فيوضحه شكل ٤ – ١٠ الذي يُبَيِّن أول خطوتين في الاشتقاق . والأعداد



شكل ٤ – ١٠

الصحيحة المكتوبة فوق وأسفل سلسلة الاشتقاق (derivation string) تبيِّن – على الترتيب – مؤشرات (indices) w و v المستخدَمة لإنشاء السلسلة .

وبدراسة شكل ٤ – ١٠ جيدا نستطيع أن نرى ماذا يحدث . فنحن نريد إنشاء حل - MPC ، ولذا فيجب أن نبدأ بالسلسلة w_1 ، أي $FS \Rightarrow$. وهذه السلسلة تحتوي على S ، وبالتالي فَكَي نُوائمها (to match it) علينا أن نستخدم v_{10} أو v_{11} . في هذا المثال نستخدم v_{10} ، وذلك يستدعي السلسلة w_{10} ، وهذا يقودنا إلى السلسلة الثانية في الاشتقاق الجزئي (partial derivation) . وبالنظر إلى خطوات عدة أكثر نرى أن السلسلة $w_1\ w_i\ w_j \ldots$ ستكون دائما أطول من السلسلة المقابلة $v_1\ v_i\ v_j \ldots$ ، وأن السلسلة الأولى متقدمة خطوة واحدة بالضبط في الاشتقاق . والاستثناء الوحيد هو الخطوة الأخيرة ، حيث w_9 يجب أن تُطبَّق لتجعل السلسلة v- تلحق بالسلسلة w- . والحل - MPC الكامل يظهر في شكل ٤ – ١١ . وطريقة الإنشاء (construction) – مع هذا المثال – تشير إلى الخطوط العامة التي تُبْنَى على أساسها النتيجة التالية .



شكل ٤ – ١١

نظرية ٤ – ٥ :

إذا كانت $G = (V, T, S, P)$ هي أي قاعدة غير مقَيَّدة (unrestricted grammar) ، و w هي أي سلسلة في T^+ ، وكان (A, B) هو زوج المقابَلة (the correspondence pair) الذي تُنْشِئه القاعدة G والسلسلة w بالعملية (process) التي يعرضها شكل ٤ – ٨ ، فإن الزوج (A, B) يسمح بوجود حل - MPC إذا وفقط إذا كانت $w \in L(G)$.

البرهان :

يشتمل البرهان على حجة استقرائية شكلية (formal inductive argument) مبنية على أساس الأسباب المشار إليها سابقا ، ولكننا سنحذف التفاصيل هنا .

وبهذه النتيجة يمكننا الآن اختزال مسألة العضوية / الانتماء بالنسبة للغات القابلة للتعديد ارتداديا إلى مسألة " بوست " المعدَّلة للمقابلة ، وبالتالي بيان عدم قابلية المسألة الأخيرة للحسم .

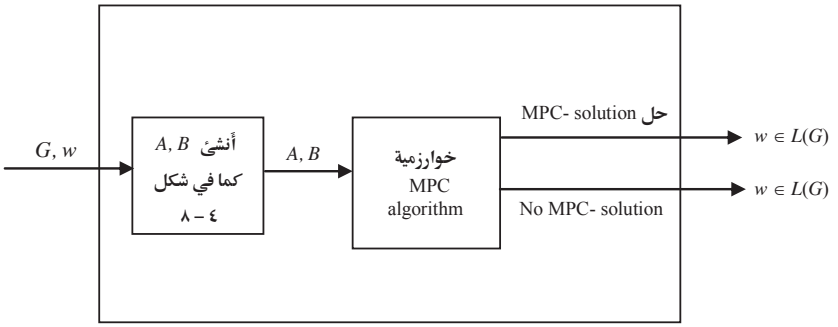
نظرية ٤ – ٦ :

مسألة " بوست " المعدَّلة للمقابَلة غير قابلة للحسم .

البرهان :

إذا أُعطينا أي قاعدة غير مقيَّدة $G = (V, T, S, P)$ وسلسلة $w \in T^+$ ، فإننا ننشئ المجموعتين A, B كما اقترحنا سابقا . وبنظرية ٤ – ٥ فإن الزوج (A, B) له حل - MPC إذا وفقط إذا كانت $w \in L(G)$.

نفرض الآن أن مسألة " بوست " المعدَّلة للمقابلة قابلة للحسم . يمكننا عندئذ إنشاء خوارزمية لمسألة انتماء G كما هو مخطط في شكل ٤ – ١٢ . ومن الواضح وجود خوارزمية لإنشاء A و B



شكل ٤ – ١٢

خوارزمية انتماء

من القاعدة G والسلسلة w ، ولكن لا توجد خوارزمية انتماء للقاعدة G والسلسلة w . وبناءً عليه فيجب أن نستنتج أنه لا يمكن وجود أي خوارزمية لحسم مسألة " بوست " المعدَّلة للمقابلة .

* * *

والآن بعد هذه النتائج الأولية نحن جاهزون لإثبات مسألة " بوست " للمقابَلة في صيغتها الأصلية .

نظرية ٤ – ٧ :

مسألة " بوست " للمقابَلة غير قابلة للحسم .

البرهان :

الحجة التي سنعتمد عليها في البرهان هي أنه لو كانت مسألة " بوست " للمقابلة قابلة للحسم لكانت مسألة " بوست " المعدَّلة للمقابلة قابلة للحسم .

نفرض أننا قد أُعطينا متتابعة $A = w_1, w_2, \ldots, w_n$ ومتتابعة $B = v_1, v_2, \ldots, v_n$ على مجموعةٍ أبجديةٍ ما Σ . سنُعَرِّف الآن رمزين جديدين Δ و $\S$ ، ومتتابعتين جديدتين

$$C = y_0, y_1, \ldots, y_{n+1}$$
$$D = z_0, z_1, \ldots, z_{n+1}$$

تُعَرَّفان كما يلي :

للقيم $i = 1, 2, \ldots, n$:

$$y_i = w_{i1}\Delta w_{i2}\Delta \ldots w_{im_i}\Delta$$
$$z_i = \Delta v_{i1}\Delta v_{i2}\Delta \ldots v_{ir_i}$$

حيث w_{ij} تعني الحرف رقم j في السلسلة w_i (jth letter of) ،

v_{ij} تعني الحرف رقم j في السلسلة v_i .

$$m_i = |\, w_i \,|\,, \mathrm{r_i} = |\, v_i \,|$$

وبالكلمات فإننا ننشئ y_i من السلسلة w_i بإلحاق (appending) **الرمز Δ بكل رمز** (character) **في السلسلة w_i (أي بوضعه بعده مباشرة) ، بينما نحصل على z_i بأن نسبق** (prefixing) **كل رمز** (character) **في السلسلة v_i بالرمز Δ (أي نضعه قبله مباشرة) .**

ولإكمال تعريف المتتابعتين C, D نأخذ

$$y_0 = \Delta y_1,$$
$$y_{n+1} = \S,$$
$$z_0 = z_1,$$
$$z_{n+1} = \Delta\S.$$

نأخذ الآن في الاعتبار الزوج (C, D) ، ونفرض أن له حلا – PC .

نظرا لوضع الرمزين $\S$, Δ (placement of) **، فإن حلا كهذا يجب أن يحتوي على y_0 على اليسار ، و y_{n+1} على اليمين ، وبالتالي يجب أن يبدو هكذا :**

$$\Delta w_{11}\,\Delta w_{12} \ldots \Delta w_{j1}\Delta \ldots \Delta w_{k1} \ldots \Delta\S = \Delta v_{11}\,\Delta v_{12} \ldots \Delta v_{j1}\Delta \ldots \Delta v_{k1} \ldots \Delta\S$$

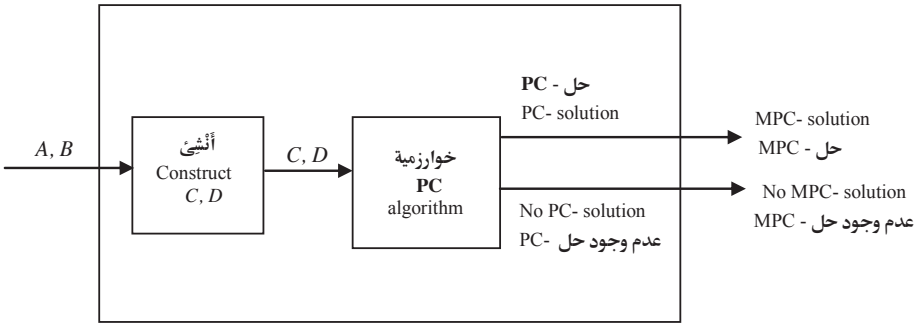
وبإهمال الرمزين $\S$,Δ ، نرى أن هذا يقتضي :

$$w_1 w_j \ldots w_k = v_1 v_j \ldots v_k$$

وهكذا فالزوج (A, B) يسمح بوجود حل - **MPC** .

ويمكننا تدوير البرهان لنثبت أنه إذا كان هناك حل - **MPC** للزوج (A, B) ، فإن هناك حلا - **PC** للزوج (C, D) .

نفرض الآن أن مسألة " بوست " للمقابلة قابلة للحسم (decidable) . يمكننا عندئذ إنشاء الآلة المبيَّنة في شكل ٤ – ١٣ . وهذه الآلة تحسم (decides) بوضوح مسألة " بوست " المعدَّلة للمقابلة . ولكن مسألة " بوست " المعدَّلة للمقابلة غير قابلة للحسم (undecidable) . وبالتالي فلا يمكننا الحصول على خوارزمية لحسم مسألة " بوست " للمقابلة .



شكل ٤ –١٣

خوارزمية **MPC**

رابعا : مسائل غير قابلة للحسم بالنسبة للغات حرة السياق

Undecidable Problems for Context - Free Languages

تُعد مسألة " بوست " للمقابَلَة أداة مناسبة لدراسة الأسئلة غير القابلة للحسم بالنسبة للغات حرة السياق . وسنوضح ذلك بإذن الله ببعض النتائج .

نظرية ٤ – ٨ :

لا توجد أي خوارزمية لتحديد / لحسم (deciding) ما إذا كانت أي قاعدة حرة السياق معطاة (any given context-free grammar) مبهمة / ملتبسة (ambiguous) أم لا .

البرهان :

نأخذ في الاعتبار متتابعتي سلاسل (two sequences of strings) :

$$A = (w_1, w_2, \ldots, w_n), \quad B = (v_1, v_2, \ldots, v_n)$$

على مجموعةٍ أبجديةٍ ما Σ **. اختر مجموعةً جديدة من رموز مختلفة** $a_1, a_2, \ldots, a_n$ **بحيث أن**

$$\{a_1, a_2, \ldots, a_n\} \cap \Sigma = \varnothing$$

وانظر إلى اللغتين

$$L_A = \{ w_i w_j \ldots w_l w_k a_k a_l \ldots a_j a_i \}$$

$$L_B = \{ v_i v_j \ldots v_l v_k a_k a_l \ldots a_j a_i \}$$

ثم انظر الآن إلى القاعدة حرة السياق

$$G = (\{S, S_A, S_B\}, \Sigma \cup \{a_1, a_2, \ldots a_n\} , P, S)$$

حيث مجموعة الإنتاجات P **هي اتحاد** (union) **مجموعتين جزئيتين : الأولى :** P_A **تتكون من**

$$S \rightarrow S_A,$$
$$S_A \rightarrow w_i S_A a_i \mid w_i a_i, \quad i = 1, 2, \ldots, n,$$

والثانية : P_B **تحتوي على الإنتاجات**

$$S \rightarrow S_B,$$
$$S_B \rightarrow v_i S_B a_i \mid v_i a_i, \quad i = 1, 2, \ldots, n.$$

خُذ

$$G_A = (\{S, S_A\}, \Sigma \cup \{a_1, a_2, \ldots a_n\} , P_A, S) ,$$

$$G_B = (\{S, S_B\}, \Sigma \cup \{a_1, a_2, \ldots a_n\} , P_B, S) .$$

من الواضح أن

$$L_A = L(G_A),$$
$$L_B = L(G_B),$$

وأن

$$L(G) = L_A \cup L_B.$$

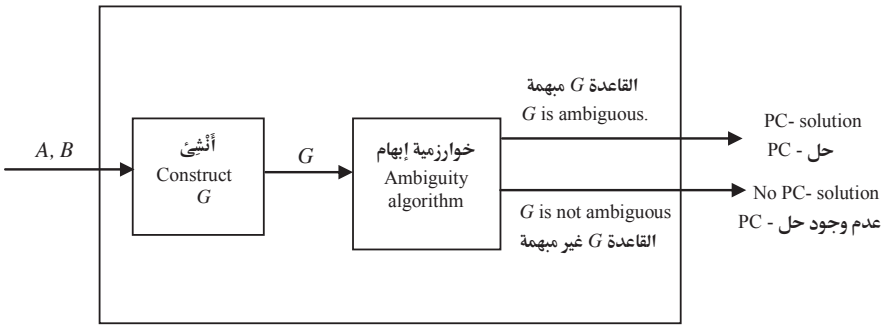
من السهل أن نرى أن القاعدتين G_A, G_B بنفسيهما غير مبهمتين (unambiguous) . وإذا انتهت سلسلة معطاة في اللغة $L(G)$ بـ a_i ، فإن اشتقاقها بالقاعدة G_A يجب أن يكون قد بدأ بـ $S \Rightarrow w_i S a_i$. وبالمثل يمكننا في أي مرحلة لاحقة أن نحدد القاعدة (rule) التي يجب أن تطبَّق . وهكذا فإذا كانت القاعدة G مبهمة ، فيجب أن تكون كذلك بسبب وجود سلسلة w لها اشتقاقان :

$$S \Rightarrow S_A \Rightarrow w_i S_A a_i \overset{*}{\Rightarrow} w_i w_j \ldots w_k a_k \ldots a_j a_i = w,$$

$$S \Rightarrow S_B \Rightarrow v_i S_B a_i \overset{*}{\Rightarrow} v_i v_j \ldots v_k a_k \ldots a_j a_i = w.$$

وبناءً عليه فإذا كانت القاعدة G مبهمة (ambiguous) ، فإن مسألة " بوست " للمقابلة مع الزوج (A, B) يكون لها حل . وبالعكس إذا كانت القاعدة G غير مبهمة ، فإن مسألة " بوست " للمقابلة لا يمكن أن يكون لها حل .

فإذا وُجدت خوارزمية لحل مسألة الإبهام / الالتباس (ambiguity problem) ، فإنه يمكننا تعديلها (adapting it) لحل مسألة " بوست " للمقابلة ، كما هو مبين في شكل ٤ – ١٤ . ولكن نظرا لأنه لا توجد خوارزمية لمسألة " بوست " للمقابلة ، فإننا نستنتج أن مسألة الإبهام / الالتباس غير قابلة للحسم .



شكل ٤ –١٤

خوارزمية PC

نظرية ٤ – ٩ :

لا توجد أي خوارزمية لتحديد / لحسم (deciding) ما إذا كانت

$$L(G_1) \cap L(G_2) = \varnothing$$

لأي قاعدتين اختياريتين حُـرَّتَـى السياق G_1, G_2 (arbitrary context - free grammars) .

البرهان :

اعتبر أن القاعدتين G_1, G_2 هما – على الترتيب – القاعدتان G_A, G_B المُعَـرِّفتان في برهان نظرية ٤ – ٨ . ونفرض أن اللغتين $L(G_A), L(G_B)$ فيهما عنصر مشترك ، أي أن

$$S_A \overset{*}{\Rightarrow} w_i w_j \ldots w_k a_k \ldots a_j a_i ,$$

$$S_B \overset{*}{\Rightarrow} v_i v_j \ldots v_k a_k \ldots a_j a_i .$$

أي أن الزوج (A, B) له حل - PC . وبالعكس إذا لم يكن للزوج حل - PC ، فلا يمكن أن يكون في اللغتين $L(G_A), L(G_B)$ عنصر مشترك . وهكذا نستنتج أن $L(G_A) \cap L(G_B)$ يكون غير خاو (nonempty) إذا وفقط إذا كان للزوج (A, B) حل - PC . وهذا الاختزال يبرهن النظرية .

خامسا سؤال عن الكفاءة

إذا كان اهتمامنا فقط بالقابلية للحوسبة (computability) أو القابلية للحسم (decidability) فإن نوع النموذج (model) الذي نستخدمه لآلة تيورنج لا يؤدي إلى اختلافات تُذكر . أما إذا بدأنا النظر إلى أمور ذات أهمية عملية كسهولة التنفيذ أو الكفاءة فإن اختلافات هامة تبدأ في الظهور . وفيما يلي مثالان لذلك .

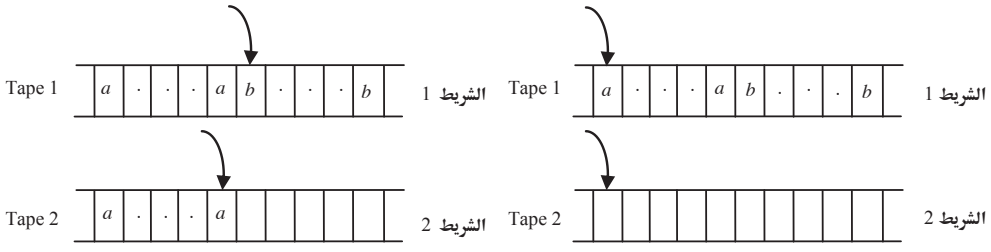
مثال ٤ – ٧ :

في مثال ١ – ٨ أنشأنا آلة تيورنج ذات شريط واحد للغة

$$L = \{a^n b^n : n \geq 1\}$$

إذا نظرنا إلى هذه الخوارزمية فإننا سنرى أنه بالنسبة للسلسلة $w = a^n b^n$ فإن الخوارزمية تحتاج تقريبا إلى عدد $2n$ من الخطوات لمواءمة (matching) كل a مع b المقابلة لها . ولذلك فإن العملية الحسابية (computation) كلها تستغرق $O(n^2)$ من التحركات . ولكننا كما أشرنا بعد ذلك في مثال ٢ – ٣ أنه يمكننا استخدام خوارزمية مختلفة بآلة

ذات شريطين . حيث ننسخ أولا جميع الرموز $a's$ في الشريط الثاني ، ثم نوائمها مع الرموز $b's$ الموجودة على الشريط الأول . ويبيِّن شكل ٤ – ١٥ محتويات الشريطين قبل وبعد عملية النسخ . ويمكننا إجراء عمليتي النسخ والمواءمة (copying and matching) بعدد من التحركات يساوي $\mathbf{O}(n)$ ، أي بكفاءة أكبر بكثير .



(ب) الشريطان بعد نسخ الرموز $a's$
Tapes after copying of $a's$

(أ) الشريطان الابتدائيان
Initial Tapes

شكل ٤ – ١٥

مثال ٤ – ٨ :

نعلم من دراسة مسألة العضوية / الانتماء (membership problem) بالنسبة للغات حرة السياق أننا إذا أخذنا طول سلسلة الإدخال w على أنه سعة / حجم المسألة (problem size) n فإن خوارزمية البحث الشامل (exhaustive search) تستغرق عدد $O(n^M)$ من الخطوات ، حيث M تعتمد على القاعدة (grammar) . وأما خوارزمية CYK الأكبر كفاءة فإنها تتطلب $O(n^3)$. وكل من هاتين الخوارزميتين تُعد خوارزمية محدَّدة (deterministic) .

وأي خوارزمية غير محددة (nondeterministic) لهذه المسألة تبدأ ببساطة بتخمين متتابعة الإنتاجات (which sequence of productions) التي تُطبَّق في عملية اشتقاق السلسلة w . فإذا استخدمنا قاعدة لا تحتوي على إنتاجات الوحدة وإنتاجات - λ (unit - and λ - productions) ، فإن طول الاشتقاق سيكون أساسا $|w|$ ، وبالتالي نحصل على خوارزمية $O(n)$.

المثالان السابقان يشيران إلى أن مسائل الكفاءة تتأثر بنوع آلة تيورنج التي نستخدمها ، وإلى أن موضوع الموازنة بين التحديد وعدم التحديد (determinism versus nondeterminism) موضوع ذو أهمية خاصة . وسنعود بإذن الله إلى هذا بتفصيل أكثر في الفصل السادس .

تمرينات رقم (٤)

أولا : بعض المسائل التي لا يمكن حلها بآلات تيورنج

٤ – ١) أثبت أن المسألة التالية غير قابلة للحسم : إذا أُعطينا أي آلة تيورنج M ، وأي رمز $a \in \Gamma$ (symbol) ، وأي سلسلة $w \in \Sigma^+$ ، فحدِّد ما إذا كان الرمز a سيُكتب إطلاقا أم لا عندما تُطبق الآلة M على السلسلة w .

٤ – ٢) اثبت أنه لا توجد خوارزمية لتقرير ما إذا كانت آلتان M_1, M_2 من آلات تيورنج تقبلان (accept) اللغة نفسها أم لا .

٤ – ٣) نفرض أن M هي أي آلة تيورنج ، وأن x, y هما وصفان لحظيان ممكنان (two possible instantaneous descriptions) للآلة M . اثبت أن مسألة تحديد ما إذا كان

$$x \vdash^*_M y$$

أم لا هي مسألة غير قابلة للحسم .

٤ – ٤) نفرض أن B هي مجموعة جميع آلات تيورنج التي تتوقف عندما نبدأها بشريط فارغ . اثبت أن هذه المجموعة قابلة للتعديد ارتداديا ، ولكنها ليست ارتدادية .

٤ – ٥) حدِّد ما إذا كانت العبارة التالية صحيحة أم لا : أيُّ مسألةٍ مجالُها (domain) محدود (finite) تكون قابلة للحسم (decidable) .

ثانيا : مسائل غير قابلة للحسم بالنسبة للغات القابلة للتعديد ارتداديا

٤ – ٦) نفرض أن M_1, M_2 آلتا تيورنج اختياريتان . اثبت أن المسألة "$L(M_1) \subseteq L(M_2)$" غير قابلة للحسم .

٤ – ٧) نفرض أن G_1 هي أي قاعدة غير مقيَّدة ، وأن G_2 هي أي قاعدة منتظمة . اثبت أن المسألة

$$L(G_1) \cap L(G_2) = \varnothing$$

غير قابلة للحسم .

٤ – ٨) نفرض أن G قاعدة غير مقيَّدة . اثبت أن السؤال " هل $L(G) = L(G)^*$ ؟ " غير قابل للحسم . استخدم في الإثبات :

أ) نظرية " رايس " .

ب) المبادئ الأولية .

ثالثا : مسألة " بوست " للمقابلة

٤ – ٩) نفرض أن $A = \{001, 0011, 11, 101\}, \quad B = \{01, 111, 111, 010\}$.
هل للزوج (A, B) حل - PC ؟ هل له حل - MPC ؟

٤ – ١٠) اثبت أنه إذا كانت $|\Sigma| = 1$ فإن مسألة " بوست " للمقابلة تكون قابلة للحسم ، أي أنه توجد خوارزمية تستطيع أن تقرر ما إذا كان للزوج (A, B) حل - PC أم لا ، وذلك لأي زوج معطَى (A, B) على مجموعة أبجدية من حرف واحد (a single - letter alphabet) .

٤ – ١١) اثبت أن التعديل التالي لمسألة " بوست " للمقابلة غير قابل للحسم :
يوجد حل - MPC إذا كانت هناك متتابعة أعداد صحيحة (a sequence of integers) بحيث أن

$$w_i w_j \ldots w_k w_1 = v_i v_j \ldots v_k v_1$$

سؤال حول الكفاءة

٤ – ١٢) نفرض أن L هي اللغة

$$L = \{ww : w \in \{a, b\}^+\}$$

ناقش كلا من إنشاء (construction) وكفاءة (efficiency) الخوارزميات لقبول اللغة على

أ) آلة تيورنج قياسية (a standard Turing machine).

ب) آلة تيورنج محدَّدة ذات شريطين (a two - tape deterministic Turing machine).

ج) آلة تيورنج غير محدَّدة ذات شريط واحد (a single - tape nondeterministic Turing machine).

د) آلة تيورنج غير محدَّدة ذات شريطين (a two - tape nondeterministic Turing machine).

الفصل الخامس

نماذج أخرى للحوسبة
Other Models of Computation

رغم أن آلات تيورنج تعد النماذج الأكثر عمومية للحوسبة (most general models of computation) التي نستطيع إنشاءها ، إلا أنها ليست الوحيدة . وقد تم اقتراح نماذج أخرى في فترات زمنية مختلفة ، وبدا بعض هذه النماذج للوهلة الأولى مختلفا اختلافا جذريا عن آلات تيورنج . إلا أننا وجدنا أخيرا أن جميع هذه النماذج متكافئة (equivalent) من حيث قدرتها على تنفيذ وإجراء الحسابات . وهذه الملاحظة يطلَق عليها عموما : " رسالة تشيرش " (**Church's thesis**) ، وهي تنص على أن جميع النماذج الممكنة للحوسبة – إذا كانت فسيحة / رحبة / واسعة بدرجة كافية (sufficiently broad) – يجب أن تكون متكافئة . كما أنها تقتضي أن هناك قيودا ذاتية / كامنة (inherent limitation) في هذا ، وأن هناك بعض الدوال التي لا يمكن التعبير عنها (expressed) بأي كيفية تعطِي طريقة صريحة (explicit method) لحسابها . وهذا الزعم (claim) مرتبط بالطبع ارتباطا وثيقا برسالة تيورنج . والربط بينهما في زعم / نظرية / مفهوم مشترك (combined notion) يطلق عليه أحيانا " رسالة تشيرش – تيورنج " (Church - Turing thesis) . وهذه تعطينا مبدأً عاما للحوسبة الخوارزمية (algorithmic computation) ، وهي – وإن كانت غير قابلة للبرهنة – تعطينا دليلا قويا على عدم إمكانية وجود نماذج أكثر قدرة (more powerful models) .

أولا : الدوال الارتدادية **Recursive Functions**

نعلم أن الدالة هي قاعدة (a rule) تسند لأي عنصر في مجموعةٍ ما – يطلَق عليها مجال **(domain)** الدالة – قيمة وحيدة في مجموعة أخرى يطلق عليها مدى **(range)** الدالة . وهذا أمر عام وواسع ، ويثير على الفور تساؤلا عن كيفية تمثيل هذا الإسناد / الارتباط (association) بصورة صريحة (explicitly) . وهناك طرق كثيرة لتعريف الدوال ، بعضها شائع الاستخدام ، والبعض الآخر أقل استعمالا . فمثلا نحن كلنا متعودون على الاصطلاح الدالي (functional notation) الذي نكتب به تعبيرا مثل

$$f(n) = n^2 + 1$$

وهـذا التعـبير يُعـرِّف الـدال f بواسـطة وَصْـفة لحسـابها : فـإذا أُعطينـا أي قيمـة للوسـيط n (argument) ، فإننا نضرب هذه القيمة في نفسها ثم نجمع واحدًا . ونظرا لأن الدالة معرَّفة بهذه الطريقة الصريحة ، فيمكننا حساب قيمها بأسلوب ميكانيكي بحت . ولإكمال تعريف الدالة f يجب علينا أيضا أن نحدد مجالها . فإذا أخذنا مثلا مجال f على أنه مجموعة جميع الأعداد الصحيحة ، فإن مدى f سيكون مجموعة جزئية من مجموعة الأعداد الصحيحة الموجبة .

ونظرا لأن كثيرا من الدوال شديدة التعقيد يمكننا تعريفها / تحديدها بهذه الطريقة ، فقد نسأل : إلى أي مدى يُعَدُّ هذا الاصطلاح (notation) عاما / شاملا (universal) ؟ بمعنى أنه إذا عَرَّفنا دالة (أي أننا نعلم العلاقة بين عناصر مجالها ومداها) ، فهل يمكننا التعبير عنها بهكذا صيغة دالية (such a functional form) ؟ وللإجابة على هذا السؤال يجب أن نوضح أولا ما هي الصيغ المسموح بها (permissible forms) . ولذلك فإننا نعرِّف بعض الدوال الأساسية (basic functions) ، مع قواعد (rules) نستخدمها لنبني من هذه الدوال الأساسية بعض الدوال الأكثر تعقيدا .

دوال ارتدادية بدائية Primitive Recursive Functions

لتبسيط الدراسة سنأخذ في الاعتبار فقط الدوال ذوات المتغير الواحد أو المتغيرين ، والتي مجالها هو إما I (وهو مجموعة جميع الأعداد الصحيحة غير السالبة) ، أو $I \times I$ ، ومداها هو I . ونبدأ بالدوال الأساسية :

(١) الدالة الصفرية $z(x)$ (The zero function)

$$z(x) = 0, \quad \forall x \in I$$

(٢) الدالة التابعة/التالية/البَعْدِيَّة/اللاحقة $s(x)$ (The successor function)

هذه الدالة قيمتها هي العدد الصحيح التالي في التتابع للعنصر x ، أي أنه بالاصطلاح المعتاد

$$s(x) = x + 1$$

(٣) دالتا الإسقاط $p_k(x_1, x_2)$ (The projector functions)

$$p_k(x_1, x_2) = x_k, \quad k = 1, 2$$

ومن هذه الدوال الأساسية يمكننا بناء دوال أكثر تعقيدا باستخدام الطريقتين التاليتين :

(أ) التركيب Composition

حيث نقوم بتركيب / بإنشاء (constructing) دالة f :

$$f(x, y) = h(g_1(x, y), g_2(x, y))$$

من دوال مُعَرَّفة g_1, g_2, h .

(ب) الارتداد البدائي Primitive recursion

حيث يمكننا تعريف دالة f ارتداديا بالعلاقتين

$$f(x, 0) = g_1(x),$$
$$f(x, y+1) = h(g_2(x, y), f(x, y))$$

من دوال مُعَرَّفة g_1, g_2, h .

وفيما يلي نوضح تطبيق ذلك ببيان كيفية بناء / إنشاء العمليات الأساسية في حسابات الأعداد الصحيحة بهذا الأسلوب .

مثال ٥ – ١ :

إضافة / جَمْع (addition) العددين الصحيحين x, y يمكن تنفيذه (implemented) بالدالة $add(x, y)$ المُعَرَّفة كما يلي :

$$add(x, 0) = x,$$
$$add(x, y+1) = add(x, y) + 1$$

فمثلا لجمع العددين الصحيحين 2, 3 نطبِّق هاتين القاعدتين (rules) تتابعيا لنحصل على :

$$\begin{aligned} add(3,2) &= add(3, 1) + 1 \\ &= (add(3, 0) + 1) + 1 \\ &= (3 + 1) + 1 \\ &= 4 + 1 = 5. \end{aligned}$$

مثال ٥ – ٢ :

باستخدام الدالة *add* المُعَرَّفة في مثال ٥ – ١ يمكننا الآن تعريف عملية الضرب (multiplication) **كما يلي :**

$$mult\ (x, 0) = 0,$$
$$mult\ (x, y+1) = add\ (x, mult\ (x, y))$$

وشكليا (formally) **فإن الخطوة الثانية هي تطبيق للارتداد البدائي ، الذي تُعَرَّف فيه *h* على أنها الدالة *add* ، بينما تُعَرَّف $g_2\ (x, y)$ على أنها دالة الإسقاط** $p_1(x, y)$. (projector function)

مثال ٥ – ٣ :

عملية الطرح (subtraction) **ليست بهذا الوضوح الكامل . فيجب أن نُعَرِّفها أولا ، آخذين في الاعتبار أن الأعداد السالبة لا يُسمح بها في نظامنا . ونُعَرِّف نوعا خاصا من الطرح بدلالة الطرح المعتاد كما يلي :**

$$x \dot{-} y = x - y \quad \text{if } x \geq y,$$
$$x \dot{-} y = 0 \quad \text{if } x < y.$$

المؤثِّر / العامل " $\dot{-}$ " (operator) **[والذي يطلَق عليه أحيانا** (the monus) **بدلا من** (the minus) **] يعرِّف عملية الطرح بحيث يكون مداها هو *I* (أي مجموعة جميع الأعداد الصحيحة غير السالبة) .**

والآن نعرِّف الدالة السابقة / المقدِّمة / القَبْلِيَّة (the predecessor function) :

$$pred\ (0) = 0,$$
$$pred\ (y + 1) = y$$

ومنها نُعَرِّف دالة الطرح (the subtracting function) :

$$subtr\ (x, 0) = x,$$
$$subtr\ (x, y+1) = pred\ (subtr\ (x, y)).$$

ولإثبات أن 2 = 3 - 5 ، أي أن $subtr\ (5, 3) = 2$ نطبِّق التعريف عدة مرات ، هكذا :

$$\begin{aligned} subtr\ (5, 3) &= pred\ (subtr\ (5, 2)) \\ &= pred\ (pred\ (subtr\ (5, 1))) \\ &= pred\ (pred\ (pred\ (subtr\ (5, 0)))) \\ &= pred\ (pred\ (pred\ (5))) \\ &= pred\ (pred\ (4)) \\ &= pred\ (3) \\ &= 2. \end{aligned}$$

وبالكيفية نفسها يمكننا تعريف القسمة الصحيحة (integer division) ، ولكننا سنترك ذلك كتدريب للقارئ الكريم . وهكذا نرى أن العمليات الحسابية الأساسية يمكن إنشاؤها (constructible) / بناؤها جميعها بالعمليات البسيطة (elementary processes) المذكورة . وبتعريف العمليات الجبرية (algebraic operations) تعريفا دقيقا (precisely defined) يمكننا الآن إنشاء عمليات أخرى أكثر تعقيدا ، وبناء / إجراء عمليات حسابية معقَّدة جدا من العمليات البسيطة . والدوال التي يمكن إنشاؤها بهذه الكيفية يُطلَق عليها " دوال ارتدادية بدائية " (primitive recursive functions) .

تعريف ٥ – ١ :

يطلَق على أي دالة إنها " ارتدادية بدائية " (**primitive recursive**) إذا وفقط إذا أمكن بناؤها / إنشاؤها (constructed) من الدوال الأساسية $\mathbf{z}, \mathbf{s}, \mathbf{p_k}$ بالتركيب المتتابع (successive composition) والارتداد البدائي (primitive recursion) .

لاحظ أنه إذا كانت g_1, g_2, h " دوال كُلِّيَّة " (total functions) ، فإن الدالة f التي تُعَرَّف بالتركيب والارتداد البدائي ستكون أيضا دالة كلية . وبناءً على هذا فإن أي دالة ارتدادية بدائية هي أيضا دالة كلية على I أو على $I \times I$.

والقدرة التعبيرية (expressive power) للدوال الارتدادية البدائية قدرة عالية / معتبرة (considerable) ، والدوال الأكثر شيوعا (most common functions) هي دوال ارتدادية بدائية . إلا أنه ليست جميع الدوال تقع في هذه الطبقة (class) ، كما توضح ذلك النظرية التالية .

نظرية ٥ – ١ :

إذا كانت F هي مجموعة جميع الدوال من I إلى I ، فإنه توجد في المجموعة F دالة ليست ارتدادية بدائية .

البرهان :

يمكننا وصف أي دالة ارتدادية بدائية بسلسلة محدودة (a finite string) تبيِّن لنا كيف تُعَرَّف هذه الدالة . ونظرا لأنه يمكن تشفير (encoding) مثل هذه السلاسل ، وتنظيمها وترتيبها (arranging) بترتيب قياسي (in standard order) ، فلذلك نرى أن مجموعة جميع الدوال الارتدادية البدائية قابلة للعد (countable) .

نفرض الآن أن مجموعة جميع الدوال هي أيضا قابلة للعد . يمكننا عندئذ كتابة جميع الدوال بترتيبٍ ما ، وليكن $f_1, f_2, \ldots$. ثم نُنشئ (construct) بعد ذلك دالة g تُعرَّف كما يلي :

$$g(i) = f_i(i) + 1, \quad i = 1, 2, \ldots$$

من الواضح أن الدالة g معَرَّفة جيدا (well defined) ، ولذلك فهي عنصر في المجموعة F ، ولكن من الواضح أيضا أن الدالة g مختلفة عن أي دالة f_i في الموضع القطري (diagonal position) . وهذا التناقض (contradiction) يثبت أن المجموعة F لا يمكن أن تكون قابلة للعد (countable) .

والجمع بين هاتين الملاحظتين يثبت أنه يجب أن توجد دالةٌ ما في المجموعة F ليست ارتدادية بدائية . وهذا هو المطلوب إثباته .

* * *

وفي الحقيقة هناك ما هو أبعد من ذلك . إذ ليس فقط : " هناك دوال ليست ارتدادية بدائية " ، بل أيضا : " هناك دوال قابلة للحوسبة (computable functions) وليست ارتدادية بدائية " .

نظرية ٥ – ٢ :

إذا كانت C هي مجموعة جميع الدوال الكلية القابلة للحوسبة (total computable functions) من I إلى I ، فإنه توجد في المجموعة C دالة ليست ارتدادية بدائية .

البرهان :

من برهان النظرية السابقة نرى أن مجموعة جميع الدوال الارتدادية البدائية قابلة للعد . نفرض أننا سنرمز لدوال هذه المجموعة بالرموز $r_1, r_2, \ldots$ ، ونُعَرِّف دالة g كما يلي :

$$g(i) = r_i(i) + 1$$

الدالة g – بطريقة إنشائها – مختلفة عن أي دالة r_i ، ولذلك فالدالة g ليست ارتدادية بدائية . ولكن من الواضح أن الدالة g قابلة للحوسبة ، مما يثبت النظرية .

* * *

البرهان غير الإنشائي / غير البنائي (nonconstructive proof) على أنه توجد دوال قابلة للحوسبة وليست ارتدادية بدائية يُعَدُّ تدريبا بسيطا نسبيا على الإقطار (diagonalization) . والبناء / الإنشاء الفعلي (actual construction) لمثال لهذه الدوال يُعَدُّ مسألة أكثر تعقيدا . وسنعطِي فيما يلي مثالا واحدا يبدو بسيطا جدا ، إلا أن إثبات أن دالة هذا المثال ليست ارتدادية بدائية طويل جدا .

دالة " أَكِّرْمان " (Ackermann's Function)

دالة " أَكِّرْمان هي دالة من $I \times I$ إلى I مُعَرَّفة كما يلي :

$$\begin{aligned} A(0, y) &= y + 1, \\ A(x,0) &= A(x - 1, 1), \\ A(x, y + 1) &= A(x - 1, A(x, y)). \end{aligned}$$

ليس من الصعب أن نرى أن الدالة A دالة كُلِّيَّة قابلة للحوسبة . وفي الحقيقة يمكننا ببساطة أن نكتب برنامجا حاسوبيا ارتداديا (a recursive computer program) لحساب الدالة A . ولكن رغم البساطة الظاهرة لدالة " أَكِّرْمان " ، إلا أنها ليست ارتدادية بدائية .

وبالطبع لا يمكننا إثبات ذلك مباشرة من تعريف الدالة A . ورغم أن هذا التعريف ليس في الصيغة المطلوبة لدالة ارتدادية بدائية ، إلا أنه من الممكن وجود تعريف بديل مناسب . والوضع هنا شبيه بذلك الذي واجهناه حين حاولنا إثبات أن لغةً ما ليست منتظمة أو ليست حرة السياق . ونحتاج هنا إلى أن ننظر إلى خاصيةٍ عامة (some general property) من خواص

طبقة جميع الدوال الارتدادية البدائية ، ونثبت أن دالة " أَكِّرْمان " لا تحقق هذه الخاصية . وبالنسبة للدوال الارتدادية البدائية ، فإن إحدى هذه الخواص هي " مُعَدَّل النمو " (growth rate) . ومعلوم أن هناك حَدًّا (a limit) لمدى السرعة (how fast) التي يمكن أن تنمو بها (can grow) أي دالة ارتدادية بدائية $f(n)$ عندما تؤول n إلى اللانهاية $n \to \infty$ ، ودالة " أَكِّرْمان " تنتهك هذا الحد (violates this limit) وتتجاوزه . ويمكننا بسهولة بيان كيف أن دالة أَكِّرْمان تنمو بسرعة كبيرة جدا (grows very rapidly) [انظر مثلا السؤال رقم ٥ – ٣ في نهاية هذا الفصل] . والعلاقة بين هذا وبين حد النمو (limit of growth) بالنسبة للدوال الارتدادية البدائية تُبَيِّنُها بدِقة النظرية التالية . وسنحذف برهان هذه النظرية نظرا لأنه طويل وممل .

نظرية ٥ – ٣ :

إذا كانت f هي أي دالة ارتدادية بدائية ، فإنه يوجد عدد صحيح ما n ، بحث أن

$$f(i) < A(n, i), \qquad \forall \; i = n, n+1, \ldots$$

نظرية ٥ – ٤ :

دالة " أَكِّرْمان " ليست ارتدادية بدائية .

البرهان :

سنبرهن النظرية بإذن الله بالتناقض (by contradiction) . نفرض أن g هي الدالة

$$g(i) = A(i, i)$$

ونفرض أن الدالة A ارتدادية بدائية . وبالتالي ستكون الدالة g أيضا ارتدادية بدائية . وبناءً عليه – بنظرية ٥ – ٣ – يوجد عدد صحيح n ، بحيث أن

$$g(i) < A(n, i) \qquad \forall \; i$$

فإذا قمنا الآن باختيار $i = n$ ، فإننا نحصل على التناقض

$$\begin{aligned} g(n) &= A(n, n) \\ &< A(n, n) \end{aligned}$$

وهذا يثبت أن الدالة A لا يمكن أن تكون ارتدادية بدائية .

الدوال الارتدادية – μ μ - Recursive Functions

لتوسيع فكرة الدوال الارتدادية حتى تشمل دالة " أَكِرْمان " ودوال أخرى قابلة للحوسبة ، يجب أن نضيف شيئا ما إلى القواعد (rules) التي يمكننا بها إنشاء مثل هذه الدوال . وإحدى الطرق للوصول إلى ذلك هي أن نقوم بتقديم المؤثِّر / المعامل μ الذي نطلق عليه " مؤثِّر الأصغريَّة " (μ or minimalization operator) ، والذي يُعَرَّف كما يلي :

$\mu y(g(x, y))$: هي أصغر قيمة لـ y بحيث أن $g(x, y) = 0$

$$[\quad \mu y(g(x, y)) = \text{smallest } y \text{ such that } g(x, y) = 0 \quad]$$

وفي هذا التعريف نفترض أن g دالة كليَّة .

مثال ٥ – ٤ :

نفرض أن

$$g(x, y) = x + y \dot{-} 3$$

حيث g دالة كليَّة (a total function) ، إذا كانت $x \leq 3$ فإن

$$y = 3 - x$$

هي نتيجة الأصغريَّة (result of the minimalization) ، بينما إذا كانت $x > 3$ فإنه لا توجد $y \in I$ بحيث أن $x + y - 3 = 0$. ولذلك فإن

إذا كانت $x \leq 3$: $\mu y(g(x,y)) = 3 - x$

إذا كانت $x > 3$: $=$ غير مُعَرَّفة (undefined)

ومن هذا نرى أنه حتى لو كانت $g(x, y)$ دالة كليَّة ، فإن $\mu y(g(x,y))$ قد تكون فقط جزئية (partial) .

كما يوضح لنا المثال السابق (مثال ٥ – ٤) فإن عملية الأصغرية (minimalization operation) تفتح الطريق أمام إمكانية تعريف دوال جزئية (partial functions) ارتداديا . ولكننا نجد كذلك أن هذه العملية توسِّع / تزيد القدرة (extends the power) على تعريف دوال كلية (total functions) لتشمل (include) جميع الدوال القابلة للحوسبة (all computable functions) . ومرة أخرى نكتفي بذكر النتيجة الرئيسية دون البرهان .

تعريف ٥ – ٢ :

يقال لدالة إنها ارتدادية - μ **(**μ**- recursive function)** إذا أمكن بناؤها / إنشاؤها (can be constructed) من الدوال الأساسية (basis functions) بمتتابعة (a sequence) من التطبيقات للمؤثر μ (applications of the μ - operator) وعمليتي التركيب والارتداد البدائي (operations of composition and primitive recursion) .

نظرية ٥ – ٥ :

أي دالة تكون ارتدادية - μ إذا وفقط إذا كانت قابلة للحوسبة .

وبناءً على هذه النظرية فإن الدوال الارتدادية - μ تعطينا نموذجا آخر للحوسبة الخوارزمية (algorithmic computation) .

ثانيا : نظم " بوست " Post Systems

أي نظام " بوست " (a Post system) يبدو مشابها تماما لقاعدة غير مقيَّدة (an unrestricted grammar) ، حيث يتكون من مجموعة أبجدية (an alphabet) ، وبعض قواعد الإنتاج التي يمكننا بها اشتقاق سلاسل متتابعة (successive strings) . ولكن هناك اختلافات جوهرية في طريقة تطبيق الإنتاجات (application of the productions) .

تعريف ٥ – ٣ :

يُعَرَّف نظام " بوست " $\prod$ **(A Post system)** بأنه

$$\prod = (C, V, A, P)$$

حيث

C : مجموعة محدودة من الثوابت (a finite set of constants) ، تتكون من مجموعتين متباعدتين (disjoint sets) :

C_N : ويطلق عليها " مجموعة الثوابت غير الطرفية " (nonterminal constants).

C_T : ويطلق عليها " مجموعة الثوابت الطرفية " (terminal constants) .

V : مجموعة محدودة من المتغيرات .

A : مجموعة محدودة من C^* ،ويطلق عليها " الموضوعات " (axioms) .

P : مجموعة محدودة من الإنتاجات (productions) .

والإنتاجات في أي نظام " بوست " يجب أن تحقق قيودا معينة . فيجب أن تكون صيغة (form) هذه الإنتاجات هي :

$$x_1\ V_1\ x_2\ \ldots\ V_n\ x_{n+1} \rightarrow y_1\ W_1\ y_2\ \ldots\ W_m\ y_{m+1} \qquad (1)$$

حيث $x_i, y_i \in C^* \ \ \& \ \ V_i, W_i \in V$ بشرط تحقق المطلبين التاليين :

- أي متغير يمكن أن يظهر على الأكثر مرة واحدة في الطرف الأيسر ، أي أن
  إذا كانت $i \neq j \quad V_i \neq V_j$
- وأي متغير في الطرف الأيمن يجب أن يظهر في الطرف الأيسر ، أي أن

$$\bigcup_{i=1}^{m} W_i \subseteq \bigcup_{i=1}^{n} V_i$$

نفرض أن لدينا سلسلة من الطرفيات (string of terminals) صيغتها $x_1w_1x_2w_2\ldots w_nx_{n+1}$ حيث السلاسل الجزئية (substrings) $x_1, x_2\ldots$ توائم (match) السلاسل المقابلة في العلاقة / الإنتاج (corresponding strings in) (1) ، وأيضا $w_i \in C^*$. يمكننا حينئذ أن نُعَرِّف / نماثل / نطابق (identify) $W's$ الموجودة بهذه القيم عن الرموز (substitute) ، ونعوض $w_1 = V_1, w_2 = V_2, \ldots$ في الطرف الأيمن في الإنتاج (1) . ونظراً لأن أي W هي V_i ما تظهر في الطرف الأيسر ، فلذلك تُسنَد إليها (assigned) قيمة وحيدة (a unique value) ، ونحصل بذلك على السلسلة الجديدة $y_1w_iy_2w_j\ \ldots\ y_{m+1}$. وهذه نكتبها هكذا :

$$x_1w_1x_2w_2\ \ldots\ x_{n+1} \Rightarrow\ y_1\ w_i\ y_2w_j\ \ldots\ y_{m+1}$$

وأما بالنسبة للقاعدة فيمكننا الآن أن نتكلم عن اللغة التي نشتقها (derive) بنظام " بوست " (by a Post system) .

تعريف ٥ – ٤ :

اللغة التي يولدها نظام " بوست " $\prod = (C, V, A, P)$ هي :

$$L(\Pi) = \left\{ w \in C_T^* : w_0 \overset{*}{\Rightarrow} w \text{ for some } w_0 \in A \right\}$$

مثال ٥ – ٥ :

نفرض أن لدينا نظام " بوست " حيث

$$\begin{aligned} C_T &= \{a, b\}, \\ C_N &= \varnothing, \\ V &= \{V_1\}, \\ A &= \{\lambda\}, \end{aligned}$$

والإنتاج

$$V_1 \to aV_1b.$$

هذا يسمح لنا بالاشتقاق

$$\lambda \Rightarrow ab \Rightarrow aabb.$$

في الخطوة الأولى نطبق الإنتاج (1) مستخدمين التعريف

$$x_1 = \lambda,\ V_1 = \lambda, \quad x_2 = \lambda,\ y_1 = a, \quad W_1 = V_1, \quad y_2 = \mathrm{b}.$$

وفي الخطوة الثانية نعيد تعريف $V_1 = ab$ ، ونترك كل شئ آخر كما هو . وإذا استمرت الخطوات هكذا ، فستقتنع بإذن الله سريعا أن اللغة التي يولدها نظام " بوست " المعطَى في هذا المثال هي : $\{a^n b^n : n \geq 0\}$.

مثال ٥ – ٦ :

نفرض أن لدينا نظام " بوست " حيث

$$C_T = \{1, +, =\},$$
$$C_N = \varnothing,$$
$$V = \{V_1, V_2, V_3\},$$
$$A = \{1 + 1 = 11\},$$

والإنتاجات

$$V_1 + V_2 = V_3 \rightarrow V_1 1 + V_2 = V_3 1,$$
$$V_1 + V_2 = V_3 \rightarrow V_1 + V_2 1 = V_3 1.$$

هذا النظام يسمح بالاشتقاق

$$1 + 1 = 11 \Rightarrow 11 + 1 = 111$$
$$\Rightarrow 11 + 11 = 1111.$$

وبتفسير سلاسل الآحاد 1's على أنها تمثيلات أحادية للأعداد الصحيحة (unary representations of integers) ، يمكننا كتابة الاشتقاق هكذا :

$$1 + 1 = 2 \Rightarrow 2 + 1 = 3 \Rightarrow 2 + 2 = 4$$

اللغة التي يولدها نظام " بوست " هذا هي مجموعة جميع متطابقات عمليات جمع الأعداد الصحيحة (identities of integer additions) مثل $2 + 2 = 4$ المشتقة من الموضوعة $1 + 1 = 2$ (axiom) .

* * *

يصوِّر لنا مثال ٥ – ٦ بطريقة مبسطة الهدف الأصلي لنظم " بوست " كآلية (mechanism) لإثبات عبارات رياضية من مجموعة من الموضوعات بطريقة محكمة . كما يبيِّن لنا العيب الذاتي لمثل هذه الطرق المحكمة تماما ، وعدم ملاءمتها ، ولماذا تُستخدم نادرا . ولكن نظم " بوست " – رغم أنها مملة في إثبات النظريات المعقدة – إلا أنها تُعَدُّ نماذج عامة للحوسبة ، كما تبيِّن ذلك النظرية التالية .

نظرية ٥ – ٦ :

أي لغة تكون قابلة للتعديد ارتداديا (recursively enumerable) إذا وفقط إذا وُجِد نظام " بوست " يولِّدها .

البرهان :

نلخص هنا خطوات البرهان وهي بسيطة نسبيا .

أولا : نظرا لأن أي اشتقاق (a derivation) بأي نظام " بوست " هو آلي / ميكانيكي تماما (completely mechanical) ، فيمكن تنفيذه بآلة تيورنج . ولذلك فإن أي لغة يولدها نظام " بوست " هي قابلة للتعديد ارتداديا .

ثانيا : بالنسبة للعكس : تَذَكَّر أن أي لغة قابلة للتعديد ارتداديا تولِّدها قاعدة ما غير مقيَّدة G (some unrestricted grammar) ، لها إنتاجات جميعها صيغتها

$$x \rightarrow y$$

حيث $x, y \in (V \cup T)^*$. وإذا أُعطينا أي قاعدة غير مقيَّدة G ، فإننا ننشئ نظام " بوست " $\prod = (V_\prod, C, A, P_\prod)$ ، حيث $V_\prod = \{V_1, V_2\}, C_N = V, C_T = T, A = \{S\}$ ، والإنتاجات (productions) :

$$V_1 x V_2 \rightarrow V_1 y V_2$$

لكل إنتاج $x \rightarrow y$ في القاعدة . ومن السهل بعد ذلك أن نثبت أن أي سلسلة w يمكن أن نولِّدها بنظام " بوست " $\prod$ إذا وفقط إذا كانت في اللغة التي تولدها القاعدة G .

ثالثا : نظم إعادة الكتابة Rewriting Systems

القواعد (grammars) المختلفة التي درسناها تتفق في عدد من الأشياء مع نظم " بوست " ، منها : أنها جميعا مَبْنِيَّة على أساس مجموعة أبجدية تُكتب فيها السلاسل ، وبعض القواعد (rules) التي يمكن بواسطتها الحصول على سلسلة من سلسلة أخرى . وحتى أي آلة تيورنج يمكن أن ننظر إليها بهذه الطريقة ، نظراً لأن وَصْفها اللحظي (instantaneous description) عبارة عن سلسلة تُعَرِّف تماما هيئتها التكوينية / الشكلية (configuration) . والبرنامج هو عندئذ مجرد مجموعة من القواعد (rules) لإنتاج سلسلة كهذه من سلسلة سابقة . هذه الملاحظات يمكن صياغتها شكليا (formalized) في مفهوم " نظام إعادة الكتابة " **(a rewriting system)** . وعموما يتكون أي نظام إعادة كتابة من مجموعة أبجدية Σ ، ومجموعة من القواعد (rules) أو إنتاجات التي يمكن بواسطتها لأي سلسلة في Σ^+ أن تُنتج سلسلة أخرى . وما يميِّز نظام إعادة كتابة عن نظام آخر هو طبيعة Σ وقيود تطبيق الإنتاجات .

والفكرة واسعة جدا وتسمح بأي عدد من حالات معيَّنَة بالإضافة إلى الحالات التي تَعَرَّضنا لها فعلا . وفيما يلي نَعْرِض باختصار بعض الحالات الهامة الأقل اشتهارا ، والتي تعطينا أيضا نماذج عامة للحوسبة .

القواعد المصفوفية Matrix Grammars

تختلف القواعد المصفوفية عن القواعد التي درسناها سابقا [والتي يُطلق عليها عادة " القواعد ذوات البِنْيَة العباراتية / الجُملية " (phrase – structure grammars)] في كيفية إمكانية تطبيق الإنتاجات . فبالنسبة للقواعد المصفوفية تتكون مجموعة الإنتاجات من مجموعات جزئية $P_1, P_2, \ldots, P_n$ كل منها عبارة عن متتابعة مرتَّبة (an ordered sequence)

$$x_1 \to y_1, x_2 \to y_2, \ldots.$$

وحينما نطبق الإنتاج الأول في مجموعةٍ ما P_i ، فيجب أن نطبق بعد ذلك الإنتاج الثاني على السلسلة التي تم إنشاؤها للتو (just created) ، ثم الإنتاج الثالث ، وهكذا . ولا يمكننا تطبيق الإنتاج الأول في مجموعةٍ P_i إلا إذا كان من الممكن أيضا تطبيق جميع الإنتاجات الأخرى في هذه المجموعة .

مثال ٥ – ٧ :

نفرض أن لدينا القاعدة المصفوفية

$$P_1 : S \to S_1S_2,$$
$$P_2 : S_1 \to aS_1,\ S_2 \to bS_2c,$$
$$P_3 : S_1 \to \lambda,\ S_2, \to \lambda.$$

أحد الاشتقاقات باستخدام هذه القاعدة هو :

$$S \Rightarrow S_1S_2 \Rightarrow aS_1bS_2c \Rightarrow aaS_1bbS_2\text{cc} \Rightarrow aabbcc.$$

لاحظ أنه كلما استخدمنا القاعدة (rule) الأولى في P_2 لإنشاء رمز a ، فإن القاعدة الثانية أيضا يجب أن تُستخدم مُنتجةً رمزين مقابلين b, c (corresponding) . وهذا يجعل من

السهل علينا أن نرى أن مجموعة السلاسل الطرفية (terminal strings) التي تولدها هذه القاعدة المصفوفية هي :

$$L = \{a^n b^n c^n : n \geq 0\}$$

* * *

والقواعد المصفوفية تحتوي على القواعد ذوات البنية العباراتية كحالة خاصة تحتوي فيها أي مجموعة P_i على إنتاج واحد بالضبط . وكذلك نظرا لأن القواعد المصفوفية تمثل عمليات خوارزمية (algorithmic processes) ، فلذلك تحكمها رسالة " تشيرش " (church's thesis) . ومن هذا نستنتج أن القواعد المصفوفية والقواعد ذوات البنية العباراتية لهما القدرة نفسها (same power) كنماذج للحوسبة (models of computation) . ولكن – كما يبيِّن لنا مثال ٥ – ٧ – استخدام قاعدة مصفوفية يعطينا أحيانا حلا أبسط بكثير مما يمكننا الوصول إليه بقاعدة ذات بنية عباراتية غير مقيَّدة .

خوارزميات " ماركوف " Markov Algorithms

خوارزمية " ماركوف " (a Markov Algorithm) هي نظام إعادة كتابة (a rewriting system) تُعتبر إنتاجاته (productions) إنتاجات مرتبة (ordered) . وفي أي اشتقاق (a derivation) يجب أن نستخدم أول إنتاج قابل للتطبيق (first applicable production) . وزيادة على ذلك يجب أن نستبدل y بالسلسلة الجزئية x (substring) التي تظهر أقصى اليسار . وقد نُفرد (single out) بعض الإنتاجات على أنها إنتاجات طرفية (terminal productions) ، وسوف نَعرض (show) هذه الإنتاجات كما يلي

$$x \rightarrow .y.$$

ويبدأ أي اشتقاق بسلسلةٍ ما $w \in \Sigma$ ، ويستمر (continues) إما إلى أن نستخدم إنتاجا طرفيا ، أو إلى ألا توجد إنتاجات قابلة للتطبيق .

وبالنسبة لقبول اللغة ، فإننا نُعَرِّف مجموعة $T \subseteq \Sigma$ من الطرفيات . ومبتدئين بسلسلةٍ طرفية (a terminal string) نطبق عددا من الإنتاجات حتى تنتج السلسلة الخاوية λ .

تعريف ٥ – ٥ :

إذا كانت M هي خوارزمية " ماركوف " (a Markov algorithm) حيث Σ هي المجموعة الأبجدية ، و T هي مجموعة الطرفيات ، فإن المجموعة

$$L(M) = \left\{ w \in T^* : w \overset{\bullet}{\Rightarrow} \lambda \right\}$$

هي اللغة التي تقبلها M .

مثال ٥ – ٨ :

نفرض أن لدينا خوارزمية " ماركوف " حيث $\Sigma = T = \{a, b\}$ وفيها الإنتاجان

$$ab \rightarrow \lambda,$$
$$ba \rightarrow \lambda.$$

نلاحظ أن أي خطوة في الاشتقاق ستحذف / ستلغي (annihilates) سلسلةً جزئية : إما ab أو ba ، وبالتالي نجد أن

$$L(M) = \{w \in \{a, b\}^* : n_a(w) = n_b(w)\}$$

مثال ٥ – ٩ :

أوجد خوارزمية " ماركوف " (a Markov algorithm) للغة

$$L = \{a^n b^n : n \geq 0\}$$

الحل :

إحدى الإجابات / الخوارزميات هي :

$$ab \rightarrow S,$$
$$aSb \rightarrow S,$$
$$S \rightarrow .\lambda.$$

في هذا المثال الأخير إذا أخذنا الإنتاجين الأَوَّلَين ، وعَكَسْنَا (reversed) الجانبين الأيسر والأيمن ، فإننا نحصل على قاعدة حرة السياق (context free grammar) تولِّد اللغة L . وبمفهوم أكيد فإن خوارزميات " ماركوف " تُعَدُّ ببساطة قواعد ذوات بنية عباراتية (phrase structure grammars) – تعمل من الخلف (working backward) . على أن هذا لا

يمكـن أَخْذُهُ حرفيـا للغاية (too literally) نظرا لأنه ليس من الواضح ماذا نعمل بالإنتاج الأخير . ولكن الملاحظة تعطي نقطة بداية لبرهان للنظرية التالية التي تحدد لنا قدرة خوارزميـات " ماركوف " ، والتي سنذكرها دون برهان .

نظرية ٥ – ٧ :

أي لغة تكون قابلة للتعديد ارتداديا (recursively enumerable) إذا وفقط إذا وُجِدت لها خوارزمية " ماركوف " .

نظـــم – L L - Systems

تُعَـدُّ أصـــول (origins) نظـــم – L مختلفـــة تمامـــا عمـــا قـــد نتوقعـــه . وقـد استَخدم " لِندنمـاير " (A. Lindenmayer) [مبتكِـــر (developer) نظـــم – L] هـــذه النظـــم لعمـــل نمـــاذج (models) لأنماط / نماذج النمو (growth patterns) لعمليات عضوية / حيوية معينة (certain organisms) . ونظـــم – L هي أساســـا نظـــم إعـــادة كتابـة متوازية (parallel rewriting systems) . ونعني بها أنه في كل خطوة من خطوات أي اشتقاق يجب إعادة كتابة (rewriting) كل رمز (symbol) . ولكي يكـون لهذا المفهوم معنى مقبول يجب أن تكون صيغة إنتاجات أي نظام – L هي :

$$a \rightarrow u \qquad (2)$$

حيث $a \in \Sigma,\ u \in \Sigma^*$. وعندما تعاد كتابة أي سلسلة (a string is rewritten) ، فيجب تطبيق أحد هذه الإنتاجات على كل رمز من رموز السلسلة قبل توليد (generating) السلسلة الجديدة .

مثال ٥ – ١٠ :

نفرض أن لدينا نظام – L مُعَرَّف بالمجموعة الأبجدية $\Sigma = \{a\}$ ، والإنتاج

$$a \rightarrow aa$$

إذا بدأنا بالسلسلة a ، فيمكننا أن نقوم بإجراء الاشتقاق

$$a \Rightarrow aa \Rightarrow aaaa \Rightarrow aaaaaaaa$$

ومن الواضح أن مجموعة السلاسل التي نحصل عليها بالاشتقاق هكذا هي :

$$L = \left\{ a^{2^n} : n \geq 0 \right\}$$

* * *

من المعلوم أن نظم – L التي إنتاجاتها في الصيغة (2) ليست عامة بدرجة كافية (not sufficiently general) لتساهم في إجراء جميع الحسابات الخوارزمية (all algorithmic computations) . وفي نظام – L المُوَسَّع / الممتد (extended L – system) تكون الإنتاجات في الصيغة :

$$(x, a, y) \rightarrow u$$

حيث $a \in \Sigma,\ x, y, u \in \Sigma^*$ ، مع التفسير بأن الرمز a يمكننا التعويض عنه (replaced) بالرمز u فقط إذا ظهر a كجزء من السلسلة $x\ a\ y$. ومعلوم أن مثل هذه النظم – L الممتدة / الموسَّعَة تُعَدُّ نماذج عامة للحوسبة (general models of computation) .

تمرينات رقم (٥)

أولا : الدوال الارتدادية

٥ – ١) نفرض أن لدينا التعريف التالي للدالة $greater\ (x, y)$

$$greater\ (x, y) = 1 \text{ if } x > y,$$
$$= 0 \text{ if } x \leq y.$$

اثبت أن هذه الدالة ارتدادية بدائية .

٥ – ٢) اثبت أن الدالة

$$g\ (x, y) = x^y$$

ارتدادية بدائية .

٥ – ٣) اثبت ما يلي بالنسبة لدالة " أكرمان " A :

أ) $A\ (1, y) = y + 2$

ب) $A\ (2, y) = 2\ y + 3$

٥ – ٤) نفرض أن

$$g\ (x, y) = 2^x + y - 3$$

احسب قيمة (compute)

$$\mu y\ (g\ (x, y))$$

وحدِّد مجالها (domain) .

ثانيا : نظم " بوست "

٥ – ٥) نفرض أن $\Sigma = \{a, b, c\}$. أوجد نظام " بوست " (a Post system) يولِّد اللغة $L = \{ww\}$.

٥ – ٦) نفرض أن $\Sigma = \{a\}$. ما هي اللغة التي يولدها نظام " بوست " ذو الموضوعة / المُسَلَّمَة / الفَرَضية $\{a\}$ (axiom) والإنتاج التالي ؟

$$V_1 \rightarrow V_1\, V_1$$

٥ – ٧) أوجد نظام " بوست " لإثبات (proving) متطابقات ضرب الأعداد الصحيحة (identities of integer multiplication) باستخدام الاصطلاح الأحادي (unary notation) ، ومبتدئا بالفَرَضِية $1 * 1 = 1$.

ثالثا : نظم إعادة الكتابة

٥ – ٨) أوجد قاعدة مصفوفة (a matrix grammar) لِلُّغَة

$$L = \{ww : w \in \{a, b\}^*\}$$

٥ – ٩) أوجد خوارزمية " ماركوف " (a Markov algorithm) تشتق اللغة

$$L = \{a^n b^n c^n : n \geq 1\}$$

٥ – ١٠) ما هي مجموعة السلاسل (strings) التي يولِّدها النظام - L (L- system) ذو الإنتاجين

$$a \rightarrow a,$$
$$a \rightarrow aa.$$

عندما نبدأه بالسلسلة a ؟

الفصل السادس

نظرة عامة على درجة التعقيد الحاسوبية
An Overview of Computational Complexity

نعود في هذا الفصل إلى إلقاء نظرة على درجة التعقيد الحاسوبية (computational complexity) ، وهي دراسة كفاءة الخوارزميات . ودرجة التعقيد – والتي ذكرناها باختصار في الفصل الثالث – تكمِّل القابلية للحوسبة (complements computability) بفَصْل (separating) المسائل التي يمكن حلها عمليا (in practice) عن تلك التي يمكن حلها فقط من حيث المبدأ (in principle) . وعند دراسة درجة التعقيد من الضروري تجاهل تفاصيل كثيرة مثل خصائص ومواصفات المعدات والمكوِّنات المادية (hardware) ، والبرمجيات (software)، وبِنَى المعطيات (data structures) ، والتنفيذ (implementation) ، وأن ننظر إلى الأمور الأساسية والعامة المشتركة (common and fundamental issues) . ولهذا السبب فإننا نتعامل غالبا مع تعابير حدود القيم (orders of – magnitude expressions) –. ولكن حتى مثل هذه النظرة عالية المستوى فإنها تعطينا – كما سنرى بإذن الله – نتائج مفيدة جدا .

وتقاس الكفاءة بمتطلبات الموارد (resource requirements) كالوقت / الزمن والمكان / السعة / الحيِّز (time and space) ، ولذلك فإننا نتكلم عن كل من درجة التعقيد الزمنية **(time complexity)** ، ودرجة التعقيد المكانية **(space complexity)** . وسنقتصر هنا على الحديث عن درجة التعقيد الزمنية ، والتي تُعَدُّ مقياسا تقريبيا للزمن الذي تستغرقه عملية حاسوبية خاصة (a particular computation) . وهناك نتائج كثيرة خاصة بدرجة التعقيد المكانية أيضا . ولكن درجة التعقيد الزمنية تُعَد أيسر قليلا في تقديرها ، وفي الوقت نفسه أكثر فائدة بالنسبة لنا .

ودرجة التعقيد الحاسوبية موضوع واسع وشامل ، ومعظمه يُعَد خارج مجال هذا الكتاب ، ولكن هناك بعض النتائج الهامة التي نعرضها بصورة مبسطة ، وهي تلقي بعض الضوء على طبيعة اللغات وطرق الحوسبة . وبإذن الله نُقَدِّم في هذا الفصل نظرة عامة موجزة على أهم النتائج الخاصة بدرجة التعقيد ، دون الدخول في التفاصيل ، ومع حذف بعض البراهين لصعوبتها .

أولا : كفاءة الحوسبة Efficiency of Computation

نبـدأ بإذن الله بإعطـاء مثـال محـدد . نفرض أننا قد أُعطينـا قائمـة مكونـة مـن ألف عـدد صحيـح ، وطُلِـب منـا ترتيبهـا / فَرْزهـا (sorting) بطـريقةٍ ما ، ولتكـن ترتيبـا تصاعديـا (ascending order) . الترتيب / الفَرْز (sorting) مسألة بسيطة ولكنها أساسية في علم الحاسوب . وإذا سُئلنا الآن : " كم من الوقت ستستغرق هذه المهمة (task) ؟ " فنرى مباشرة أننا نحتاج إلى معلومات أكثر قبل إمكانية الإجابة على هذا السؤال . من الواضح أن عدد العناصر في القائمة يلعب دورا هاما في تحديد الوقت المطلوب لإنجاز المهمة ، ولكن هناك عوامل أخرى . من هذه العوامل : نوع الحاسوب الذي نستخدمه ، وكيفية كتابة البرنامج الذي سينفذ هذه المهمة . كذلك هناك عدد من طرق الترتيب ، فاختيار إحدى هذه الطرق / الخوارزميات عامل هام . وهناك عوامل أخرى ، ولكننا سنكتفي هنا بهذه العوامل الأساسية التي ذكرناها .

ولتبسيـط مناقشتنـا لموضـوع درجـة التعقيـد الحاسوبيـة نذكـر الافتراضات التاليـة (simplifying assumptions) :

(١) نمـوذج (model) دراستنـا سيكـون هـو آلـة تيورنـج . وأما نـوع آلـة تيورنـج – التي سنستخدمها – بالضبط فسوف نناقشه فيما بعد .

(٢) سنرمـز لحجم / لِسِعَةِ المسألة (size of the problem) بالحرف n . وبالنسبة لمسألتنا الحاليـة : مسـألة الترتيـب (sorting problem) واضـح أن n هـي عـدد العناصـر (items) في القائمـة (list) . ورغم أن حجم أي مسألة ليس من السهل دائما تحديده / توصيفه (characterized) ، إلا أنه يمكننا عموما جعله مرتبطا / متعلقا – بطريقةٍ ما (in some way) – بعدد صحيح موجب .

(٣) عند تحليل أي خوارزمية يكون اهتمامنا الأكبر بسلوكها العام (general behavior) ، وليس بأدائها (performance) في حالةٍ بعينها (a specific case) . وبصورة خاصة فإننا نهتم بكيفية سلوك الخوارزمية عندما يكبر / يزداد حجم المسألة (size increases) . وبسبب هذا فإن السؤال الأساسي يتضمن السؤال عن مدى سرعة (how fast) نمو (growth) / زيادة الوقت المطلوب والحيِّز المطلوب (time and space requirements) عندما تصبح قيمة n كبيرة .

فهدفنـا الأول إذًا هـو تحديـد الوقـت المطلـوب – لحـل مسألة معينـة – كدالـةٍ فـي حجمهـا (function of its size) باستخـدام آلـة تيورنـج باعتبـارها النمـوذج الحـاسوبي (computer model) .

نعطِي أولا معنى مفهوم الوقت بالنسبة لآلة تيورنج. فنحن ننظر إلى آلة تيورنج على أنها آلة تقوم بتنفيذ حركة واحدة في وحدة الزمن (making one move per time unit) . وبالتالي فإن الوقت المستغرق لإجراء عملية حاسوبية (a computation) هو عدد الحركات التي تم تنفيذها (number of moves made) . وكما ذكرنا فإننا نريد دراسة كيفية زيادة المتطلبات الحاسوبية مع زيادة حجم المسألة . وعادةً في مجموعة جميع المسائل ذوات حجم معطَى (a given size) فإننا نجد اختلافا وتنوعا (a variation) . وهنا نهتم فقط بأسوأ حالة **(worst case)** التي تكون فيها المتطلبات أعلى ما يمكن (highest source requirements) . فإذا قلنا إن عملية حاسوبية معينة (a computation) تحتاج لدرجة تعقيد زمنية $T(n)$ (time – complexity) فإننا نعني أن العملية الحاسوبية لأي مسألة حجمها n سيمكن إتمامها (completed) بعدد من الحركات (moves) لا يزيد عن (in no more than) $T(n)$ على آلة تيورنج ما .

وبعد الاتفاق على نوع معين من آلات تيورنج كنموذج حاسوبي ، فإنه يمكننا تحليل الخوارزميات بكتابة برامج صريحة وحساب عدد الخطوات المطلوبة لحل المسألة . ولكن لأسباب عدة لا يفيدنا هذا كثيرا . فأولا : عدد العمليات (operations) التي يتم تنفيذها قد يتغير مع تغير التفاصيل البسيطة في البرنامج ، وبالتالي قد يعتمد بشدة على المبرمج . وثانيا : من وجهة نظر عملية نحن يهمنا كيفية أداء الخوارزمية مهمتها في دنيا الحقيقة والتي قد تختلف كثيرا عن كيفية أدائها على آلة تيورنج . وأفضل ما يمكن أن نطمح إليه هو أن يكون تحليل (analysis) آلة تيورنج ممثِّلا (representative) للخصائص الرئيسية (major aspects) للأداء الفعلي في الواقع العملي (real – life performance) ، مثل مُعَدَّل التزايد / النمو التقاربي (asymptotic growth rate) لدرجة التعقيد الزمنية . وأول محاولة سنقوم بها لفهم متطلبات الموارد (resource requirements) لأي خوارزمية ستكون لذلك تحليلاً لحدود القيم (an order – of – magnitude analysis) نستخدم فيه الاصطلاحات[*] O, θ, Ω . ورغم أن هذه الطريقة غير رسمية (informal) ، إلا أنها تأتي عادة بنتائج ومعلومات مفيدة .

(*) انظر مثلا : اصطلاحات حدود القيم / اصطلاحات التقارب .. الفصل الأول .. كتاب " هياكل البيانات بلغة C++ " .. حمزة سيد رشوان وأبو بكر أحمد السيد .. مكتبة الفلاح للنشر والتوزيع .. الكويت .. ٢٠٠٦

مثال ٦ – ١ :

نفرض أننا قد أُعطينا مجموعة من الأعداد $x_1, x_2, \ldots, x_n$ عددها n ، وكذلك عددا مفتاحا x (a key number) . حدد ما إذا كانت المجموعة المعطاة S تحتوي على x أم لا .

إذا لم تكن المجموعة S مرتبة بطريقةٍ ما ، فإن أبسط خوارزمية هي مجرد بحث خطي **(a linear search)** نقوم فيه بمقارنة x على التوالي / التتابع (successively) بالعناصر $x_1, x_2, \ldots$ إلى أن نجد توافقا (a match) أو إلى أن نصل إلى آخر عنصر في المجموعة S . ونظرا لأننا قد نجد توافقا في أول مقارنة أو في آخر مقارنة أو لا نجد توافقا على الإطلاق ، فإننا لا نستطيع التنبؤ بكمية / بقدر العمل (how much work) المطلوب ، ولكننا نعلم أنه في أسوأ حالة (in the worst case) علينا أن نقوم بإجراء عدد n من المقارنات . يمكننا عندئذ أن نقول إن درجة التعقيد الزمنية لهذا البحث الخطي هي $\mathbf{O}(n)$ ، أو حتى أفضل من هذا $\boldsymbol{\theta}(n)$. وفي هذا التحليل لم نقم بذكر أي افتراضات معينة (specific assumptions) عن ماهية الآلة (what machine) التي نفَّذنا عليها هذه الخوارزمية ، أو عن كيفية تنفيذ الخوارزمية . ولكننا نرى أنه إذا ضاعفنا حجم مجموعة الأعداد فإن زمن العملية الحاسوبية (computation time) سيتضاعف تقريبا . وهذا يخبرنا كثيرا عن عملية البحث .

ثانيا : نماذج آلات تيورنج ودرجة التعقيد
Turing Machine Models and Complexity

عند دراسة القابلية للحوسبة (computability) لا يؤدي استخدام نموذج معين لآلة تيورنج إلى اختلافات تُذْكَر ، ولكننا رأينا سابقا أن كفاءة أي عملية حاسوبية يمكن أن تتأثر بعدد شرائط الآلة (tapes of the machine) ، وبما إذا كانت الآلة محدَّدة (deterministic) أو غير محدَّدة (nondeterministic) . وكما بَيَّن مثال ٤ – ٨ فإن الحلول غير المحدَّدة (nondeterministic solutions) تكون عادةً أكبر كفاءة بكثير من بديلاتها المحدَّدة . والمثال التالي يوضح ذلك أيضا بصورة أوضح .

مثال ٦ – ٢ :

يعرض هذا المثال " مسألة التَحَقُّقِيَّة " **(satisfiability problem) (SAT)** التي تلعب دورا هاما في نظرية التعقُّد / التعقيد (complexity theory) .

يُعَرَّف الثابت أو المتغيِّر المنطقي أو البُولي (a logic or boolean constant or variable) بأنه ثابت أو متغير يمكنه أن يأخذ إحدى قيمتين فقط : صح أو خطأ (صادق أو كاذب) (true or false) ، وسنرمز لهاتين القيمتين بـ : 1 و 0 ، حيث القيمة 1 ترمز إلى صح / صادق / true ، والقيمة 0 ترمز إلى : خطأ / كاذب / false . وتُستخدم المؤثِّرات البُولية (boolean operators) للجمع بين (combining) ثوابت ومتغيرات بُولية في تعابير بُولية (boolean expressions) . وأبسط المؤثرات البُولية هي :

- أو : **or** ، ويُرمز له بالرمز $\vee$ ، ويُعَرَّف بما يلي :

$$0 \vee 1 = 1 \vee 0 = 1 \vee 1 = 1,$$
$$0 \vee 0 = 0.$$

- و : **and** ، ويُرمز له بالرمز $\wedge$ ، ويُعَرَّف بما يلي :

$$0 \wedge 0 = 0 \wedge 1 = 1 \wedge 0 = 0,$$
$$1 \wedge 1 = 1.$$

- النفي : **negation** ، ويُرمز له بالرمز [خط (bar)] ، ويُعَرَّف بما يلي :

$$\bar{0} = 1,$$
$$\bar{1} = 0.$$

والآن نأخذ في الاعتبار التعابير البُولية في الصيغة القياسية / المعيارية العَطْفِيَّة **(CNF) (conjunctive normal form)** وفي هذه الصيغة نُنشئ تعابير (expressions) من المتغيرات $x_1, x_2, \ldots, x_n$ مبتدئين بالعلاقة

$$e = t_i \wedge t_j \wedge \ldots \wedge t_k \qquad (1)$$

حيث الحدود $t_i, t_j, \ldots t_k$ (terms) يتم تكوينها / إنشاؤها (created) باستخدام المؤثِّر **or** ليجمع بين متغيرات ونفيها (negation) ، أي أن

$$t_i = s_l \vee s_m \vee \ldots \vee s_p \qquad (2)$$

حيث أي من $s_1, s_m, \ldots, s_p$ يرمز إلى متغير أو نفي متغير . والرمز s_i سيُطْلَق عليه " حَرْفِيّ " (**literal**) ، بينما الرمز t_i سيُطلق عليه " عبارة " (**clause**) : " عبارة تعبير e بالصيغة القياسية العَطْفِيَّة (**CNF**) " (clause of a CNF expression e) .

ومسألة التَحَقُّقِيَّة (satisfiability problem) يمكننا الآن صياغتها ببساطة كما يلي: إذا أُعطيت تعبيرا تَحَقُّقِيا e (a satisfiable expression) في الصيغة القياسية العَطْفية CNF ، فأوجد إسنادا لقيم (assignment of values) المتغيرات $x_1, x_2, \ldots, x_n$ بحيث يجعل قيمة التعبير e " صح " (true) . وكحالة خاصة افرض أن

$$e_1 = (\bar{x}_1 \vee x_2) \wedge (x_1 \vee x_3)$$

إسناد القيم $x_1 = 0, x_2 = 1, x_3 = 1$ (the assignment) يجعل قيمة التعبير e_1 " صح " (true) ، وبالتالي فهذا التعبير تَحَقُّقِي . ومن ناحية أخرى التعبير

$$e_2 = (x_1 \vee x_2) \wedge \bar{x}_1 \wedge \bar{x}_2 \qquad (3)$$

ليس تَحَقُّقِيًّا نظرا لأن أي إسناد لقيمتي المتغيرين x_1, x_2 سيجعل قيمة e_2 " خطأ " (false).

ومن السهل أن نكتشف خوارزمية محدَّدة (a deterministic algorithm) لمسألة التحققية . نأخذ جميع القيم المحتملة (possible values) للمتغيرات $x_1, x_2, \ldots, x_n$ ، ولكل منها نحسب قيمة التعبير (evaluate the expression) . ونظرا لأن هناك عدد 2^n من هذه الاحتمالات ، فدرجة التعقيد الزمنية (time-complexity) لهذه الطريقة الشاملة / الاستنفادية (exhaustive approach) أسِّيَّة (exponential) .

ومرة أخرى فإن الخوارزمية غير المحددة البديلة (nondeterministic alternative) تُبَسِّط الأمور . فإن كان التعبير e تحقُّقِيًّا (satisfiable) ، فإننا نخمِّن قيمة كل متغير x_i ، ثم نحسب قيمة e (evaluate) . وهذه خوارزمية درجتها أساسا $O(n)$. وكما رأينا في مثال ٤ – ٨ فإن لدينا خوارزمية بحث شاملة / استنفادية محدَّدة (a deterministic exhaustive search algorithm) درجة تعقيدها أسِّيَّة (exponential complexity)، وخوارزمية غير محددة خَطِّيَّة الزمن (a linear-time

(nondeterministic algorithm) . إلا أننا – بعكس مثال ٤ – ٨ – لا نعرف أي خوارزمية محدَّدة غير أُسِّيَّة (nonexponential deterministic algorithm) .

نظرية ٦ – ١ :

إذا فرضنا أن آلة ذات شريطين (a two-tape machine) **أمكنها أن تنفذ عملية حاسوبية** (a computation) **في عدد** n **من الخطوات** (steps) **، فإنه يمكننا محاكاة** (simulating) **هذه العملية الحاسوبية بآلة تيورنج قياسية** (standard) **في عدد** $O(n^2)$ **من الخطوات .**

البرهان :

لمحاكاة العملية الحاسوبية على الآلة ذات الشريطين ، تحتفظ (keeps) **الآلة القياسية بالوصف اللحظي** (instantaneous description) **للآلة ذات الشريطين على شريطها كما هو مبين في شكل ٦ – ١ . ولمحاكاة حركة واحدة** (one move) **، تحتاج الآلة القياسية لأن**

	□	a	b	q	b	a	a	×	a	q	b	b	b	a	□

شكل ٦ – ١

تبحث (to search) **في المنطقة النشطة (الفعالة) في شريطها بأكملها** (the entire active area of its tape) . **ولكن نظرا لأن حركة واحدة للآلة ذات الشريطين يمكن أن توسِّع** (extend) **المنطقة النشطة بخليتين** (two cells) **على الأكثر ، فبعد عدد** n **من الحركات سيكون طول المنطقة النشطة** $O(n)$ **على الأكثر . ولذلك فإن المحاكاة الكلية** (entire simulation) **يمكن أن تُنَفَّذ بعدد** $O(n^2)$ **من الحركات .**

ملاحظة :

هذه النتيجة يمكن أن نعممها (generalize) **بسهولة لأكثر من شريطين ، أي يمكننا إثبات أن عدد** n **من الحركات** (n moves) **في آلة تحتوي على عدد** k **من الشرايط** (on a k-tape machine) **يمكننا أن نحاكيها على آلة قياسية في عدد** $O(n^2)$ **من الحركات .**

نظرية ٦ – ٢ :

إذا فرضنا أن آلة تيورنج غير محددة M (a nondeterministic Turing machine) **يمكنها أن تنفذ عملية حاسوبية في عدد** n **من الخطوات ، فإن آلة تيورنج قياسية يمكنها أن تنفذ العملية الحاسوبية نفسها في عدد** $O(k^{an})$ **من الخطوات ، حيث** k **و** a **لا يعتمدان على** n (independent of) .

البرهان :

أي آلة تيورنج قياسية يمكنها محاكاة آلة غير محددة بأن تَتَتَبَّع (keep track of) **جميع التشكيلات** (configurations) **الممكنة ، باحثةً باستمرار** (continually searching) **ومُحَدِّثَةً** (updating) **المنطقة النشطة بأكملها** (entire active area) . **وإذا كان** k **هو أكبر معامل تَفَرُّع** (maximum branching factor) **لعدم التحديد** (for the nondeterminism) ، **فبعد عدد** n **من الخطوات سيوجد على الأكثر عدد** k^n **من التشكيلات الممكنة** (possible configurations) . **ونظرا لأنه يمكن إضافة** (adding) **رمز واحد على الأكثر إلى كل تشكيلة بحركة واحدة** (by a single move) ، **فلذلك يكون طول التشكيلة الواحدة بعد عدد** n **من الحركات هو** $O(n)$. **وهكذا فلكي نحاكي حركة واحدة ، فإن الآلة القياسية يجب أن تبحث في منطقة نشطة طولها** $O(nk^n)$ ، **مما يؤدي إلى النتيجة المطلوبة . وللنظر في بعض التفاصيل انظر السؤال رقم ٦ – ٣ بالتمرينات بنهاية الفصل .**

ملاحظة :

يجب أن نكون حذرين عند تفسير هذه النظرية . فهي تنص على أن أي عملية حاسوبية غير محددة يمكن دائما إجراؤها على آلة محددة إذا كنا راغبين في أن نأخذ في الاعتبار زيادة أُسِّيَّة (an exponential increase) **في الوقت المطلوب . ولكن هذا الاستنتاج يأتي من محاكاة ذات أفق بسيط خاص** (a particularly simple-minded simulation) ، **ولا نزال نأمل في أداء أفضل . واستكشاف هذا الموضوع هو لُبُّ نظرية التعقيد / التعقُّد** (complexity theory) .

ويشير مثال ٤ – ٧ إلى أن خوارزميات آلة متعددة الشرايط (a multitape machine) **قد تكون أقرب إلى ما قد نستخدمه عمليا** (in practice) **من الطريقة المرهقة** (cumbersome) **لآلة تيورنج قياسية . ولهذا السبب فسنستخدم بإذن الله آلة تيورنج متعددة الشرايط كنموذجنا** (model) **لدراسة مسائل وموضوعات التعقُّد** (complexity issues) ، **ولكن هذا كما سنرى أمر ثانوي .**

ثالثا : عائلات اللغات وطبقات التعقيد
Language Families and Complexity Classes

في التدرج الهرمي (التسلسل) لتشومسكي (Chomsky hierarchy) لتصنيف اللغات (language classification) نوجد ارتباطا (association) بين عائلات اللغات وطبقات الآلات (classes of automata) ، حيث تُعَرَّف كل طبقة آلات بطبيعة تخزينها المؤقت (nature of its temporary storage) . وكإمكانيةٍ أخرى لتصنيف اللغات نستخدم آلة تيورنج ونعتبر درجة التعقيد الزمنية (time – complexity) مقياسا / عاملا للمقارنة والتمييز (a distinguishing factor) . وكي نقوم بهذا نعـرِّف أولا درجة التعقيد الزمنية لأي لغة .

تعريف ٦ – ١ :

يُـقال إن آلة تيورنج M تَحْسِم (decides) / تُحدِّد / تُقرر / تتخذ قرارا بالقبول أو الرفض بشأن لغةٍ L في وقت / فترة زمنية $T(n)$ إذا تم اتخاذ قرار (deciding) بشأن كل كلمة w في اللغة L حيث $n = |w|$ في عدد $T(n)$ من الحركات . وإذا كانت M آلة غير محددة (nondeterministic) ، فهذا يقتضي أنه لكل سلسلة $w \in L$ توجد على الأقل متتابعة (sequence) واحدة من الحركات طولها أقل من أو يساوي $T(|w|)$ تؤدي إلى القبول (acceptance) ، وأن آلة تيورنج تتوقف (halts) على جميع المدخلات (on all inputs) في فترة زمنية $T(|w|)$.

تعريف ٦ – ٢ :

أي لغة L يُقال إنها عنصر في الطبقة $DTIME(T(n))$ إذا وُجدت آلة تيورنج محددة متعددة الشرايط (a deterministic multitape Turing machine) تحسم / تحدد اللغة L (decides) في وقت يستغرق $O(T(n))$.

وأي لغة L يُقال إنها عنصر في الطبقة $NTIME(T(n))$ إذا وُجدت آلة تيورنج غير محـددة متعـددة الشرايـط (a nondeterministic multitape Turing machine) تحسم / تحدد اللغة L (decides) في وقت يستغرق $O(T(n))$.

وهناك بعض العلاقات الواضحة بين طبقات التعقيد (complexity classes) هذه ، مثل :

$$DTIME\ (T(n)) \subseteq NTIME(T(n))$$

&

$$T_1(n) = O(T_2(n))$$

$$\Rightarrow$$

$$DTIME\ (T_1(n)) \subseteq DTIME(T_2(n))$$

ولكـن مـن هنـا يصبـح الوضـح مبهمـا وغير واضـح (obscure) بسرعـة . وما يمكننـا أن نقوله هو أنه كلما زادت حدود $T(n)$ (order of) كلما تلقينا لغات أكثر بصورة تصاعدية (progressively) . ونذكر النظرية التالية دون برهانها .

نظرية ٦ – ٣ :

لكل عدد صحيح $k \geq 1$

$$DTIME\ (n^k) \subset DTIME\ (n^{k+1})$$

* * *

والاستنتـاج الـذي يمكننـا أن نستخلصـه مـن هـذا هـو أنـه توجـد بعـض اللغات التي يمكننـا حسمها / تحـديدها (decided) (اتخاذ قرار بشأن قبولهـا) في وقـت $O\ (n^2)$ (in time) ، ولا توجـد لهـا أي خوارزميـــة انتمـــاء خطيـة الزمـن (linear – time membership algorithm)،وكذلك أنه توجد لغات في $DTIME\ (n^3)$ والتي هي ليست في $DTIME(n^2)$، وهكذا . وهذا يعطينا عددا لا نهائيا من طبقات التعقيد المتداخلة (nested complexity classes) . ونحـصل حتى على أكـثر من هذا إذا سمحنـا بتعقيد أُسِّـيٍّ الزمن (exponential time complexity) . وفي الحقيقة لا يوجد أي حد (limit) لهذا ، فَمَهْمَا كان مدى سرعة نمـو (growth) وزيـادة دالة التعقيـد (complexity function) $T(n)$ ، يوجد دائما شيءٌ ما خارج $DTIME(T\ (n)$ (outside) .

نظرية ٦ – ٤ :

لا توجد أي دالة تيورنج كُلِّيَّة قابلة للحوسبة (total Turing computable function) $f\ (n)$ بحيث أن أي لغة ارتدادية (recursive language) يمكن حسمها (decided) في وقت يساوي $f\ (n)$ ، حيث n هي طول سلسلة الإدخال (length of the input string) .

البرهان :

نأخذ في الاعتبار المجموعة الأبجدية $\Sigma = \{0, 1\}$ (alphabet) مع جميع السلاسل في Σ^+ (strings in) مُرَتَّبَة (arranged) في ترتيب مناسب / صحيح (in proper order) $w_1, w_2, \ldots$. وافرض أيضا أن لدينا ترتيبا مناسبا لآلات تيورنج $M_1, M_2, \ldots$.

نفرض الآن أن الدالة $f(n)$ المذكورة في منطوق النظرية موجودة . يمكننا عندئذ تعريف اللغة

$$L = \{w_i : M_i \text{ does not decide } w_i \text{ in } f(|w_i|) \text{ steps}\} \qquad (4)$$

أي أن اللغة L هي مجموعة السلاسل w_i التي لا تحسمها آلات تيورنج المقابلة M_i في عدد $f(|w_i|)$ من الخطوات .

نحن نَدَّعي أن اللغة L ارتدادية . وكي نرى ذلك نأخذ في الاعتبار أي سلسلة $w \in L$ ، ونحسب أولا $f(|w|)$. وهذا ممكن (possible) بفرض أن f دالة تيورنج كُلِّيَّة قابلة للحوسبة . وبعد ذلك نوجد موضع i (position) للسلسلة w في المتتابعة $w_1, w_2, \ldots$. وهذا ممكن أيضا لأن المتتابعة مرتبة ترتيبا سليما / مناسبا (in proper order) . وعندما يكون لدينا i ، فإننا نوجد M_i ونجعلها تؤثِّر على السلسلة w (operate on) لعدد $f(|w|)$ من الخطوات . وهذا سيخبرنا ما إذا كانت w في اللغة L أم لا ، وهكذا فإن L ارتدادية .

ولكن يمكننا الآن أن نبيِّن أن اللغة L ليست قابلة للحسم (decidable) في وقتٍ يساوي $f(n)$ (in time) . ولإثبات ذلك بالتناقض نفرض أن L قابلة للحسم في هذا الوقت (أي في هذه الفترة الزمنية) . نظرا لأن L ارتدادية ، فتوجد آلة M_k ما ، بحيث أن $L = L(M_k)$. هل w_k موجودة في اللغة L ؟ فإذا زعمنا أن w_k موجودة في L ، فإن M_k تحدِّد / تقرر / تحسم (decides) w_k في عدد $f(|w_k|)$ من الخطوات . ولكن هذا يناقض العلاقة (4) . وبالعكس إذا فرضنا أن $w_k \notin L$ فإننا نصل إلى تناقض . وعدم القدرة على حل هذه النقطة يُعَدُّ نتيجة إقطارية نمطية (a typical diagonalization result) ، ويؤدي بنا إلى استنتاج أن الفرض الأصلي – وهو وجود دالة قابلة للحوسبة f (computable) (n) – يجب أن يكون خاطئا .

* * *

نظرية ٦ – ٣ تسمح لنا بعرض بعض المزاعم (claims) كأن نزعم مثلا أنه توجد لغة في $DTIME\ (n^4)$ ليست موجودة في $DTIME\ (n^3)$. ورغم أن هذا قد يُعد موضوعا ذا أهمية نظرية (theoretical interest) ، إلا أنه ليس من الواضح إن كانت نتيجة كهذه لها أي مغزى عملي (practical significance) . وعند هذه النقطة ليس لدينا أي قرائن أو مؤشرات إلى ماهية الخصائص (characteristics) التي قد تتميز بها أي لغة في $DTIME\ (n^4)$. ويمكننا النظر أعمق قليلا في هذه المسألة إذا أوجدنا علاقة بين تصنيف التعقيد (complexity classification) واللغات الموجودة في التدرج الهرمي لتشومسكي (Chomsky hierarchy) . وسننظر إلى بعض الأمثلة البسيطة التي تعطينا بعض النتائج الأكثر وضوحا .

مثال ٦ – ٣ :

يمكننا التَّعَرُّف على (recognizing) أي لغة منتظمة (regular language) بآلةٍ محدودةٍ محددة (a deterministic finite automaton) في وقت يتناسب مع (proportional to) طول المدخلات (length of the input) . ولذلك فإن

$$L_{REG} \subseteq DTIME\ (n)$$

ولكن $DTIME\ (n)$ تحتوي على أكثر بكثير من L_{REG} . وقد بَيَّنَّا فعلا في مثال ٥ – ٧ أن اللغة حرة السياق $\{a^n b^n : n \geq 0\}$ يمكن التعرف عليها بآلة ذات شريطين (a two – tape machine) في وقت $O\ (n)$. والحجة (argument) التي ذكرناها هناك يمكننا أن نستخدمها حتى لِلُغات أكثر تعقيدا .

مثال ٦ – ٤ :

اللغة غير حرة السياق (non-context-free language)

$$L = \{ww : w \in \{a, b\}^*\}$$

موجودة في $NTIME\ (n)$.

هذه النتيجة تنتج مباشرةً حيث أنه يمكننا التَّعَرُّف على السلاسل (strings) الموجودة في هذه اللغة بالخوارزمية التالية :

(١) انسخ المدخلات من ملف الإدخال (input file) إلى الشريط 1 (tape) . خَمِّن (guess) بطريقة غير محدَّدَة (nondeterministically) موضع منتصف السلسلة (middle of the string) .

(٢) انسخ الجزء الثاني (second part) إلى الشريط 2 (tape) .

(٣) قارن الرموز (symbols) الموجودة على الشريط 1 بتلك الموجودة على الشريط 2 رمزًا رمزا .

من الواضح أن جميع الخطوات يمكن أن تُنَفَّذ في وقت $O(|w|)$ ، وبالتالي فإن $L \in NTIME(n)$.

وفعليا يمكننا إثبات أن $L \in DTIME(n)$ إذا استطعنا ابتكار خوارزمية لإيجاد منتصف سلسلة (middle of a string) في زمن $O(n)$ (time) . وهذا يمكن تحقيقه كما يلي :

ننظر إلى كل رمز (symbol) على الشريط tape 1 ، ونحتفظ بِعَدَّاد على الشريط tape 2 (keeping a count on)، ولكن نَعُدُّ فقط كل ثاني رمز (counting only every second symbol) . وسنترك التفاصيل كتدريب للقارئ الكريم .

مثال ٦ – ٥ :

نستنتج من مثال ٤ – ٨ أن

$$L_{CF} \subseteq DTIME(n^3)$$

وكذلك

$$L_{CF} \subseteq NTIME(n)$$

نأخذ في الاعتبار الآن عائلة اللغات الحساسة للسياق (context – sensitive languages) . الإعراب ذو البحث الشامل / المستفيض (exhaustive search parsing) يمكن إجراؤه هنا أيضا نظرا لأن عددا محدودا فقط من الإنتاجات (only a limited number of productions) يمكن تطبيقه (applicable) في كل خطوة . ويمكننا أن نرى (*) أن أكبر عدد من الصيغ العباراتية (maximum number of sentential forms) هو :

(*) انظر كتاب : نظرية الحوسبة والأوتوماتية واللغات الشكلية .. أبو بكر أحمد السيد .. مكتبة الفلاح .. الكويت 2011 ، الفصل الخامس : اللغات حرة السياق . وتتبَّع التحليل الذي أدى إلى الوصول إلى المعادلة (2) صفحة 159 .

$$N = |P| + |P|^2 + \ldots |P|^{cn} = O(|P|^{cn+1})$$

ولكن لاحظ أننا لا نستطيع أن نزعم من هذا أن

$$L_{CS} \subseteq DTIME(|P|^{cn+1})$$

وذلك لأننا لا نستطيع أن نضع حدًّا أعلى (an upper bound) على $|P|$ و c .

* * *

من الأمثلة السابقة نلاحظ أنه كلما زادت $T(n)$ ، كلما تم تغطية (covering) لغات أكثر من العائلات L_{REG}, L_{CF}, L_{CS} . ولكن الصلة (connection) بين التدرج الهرمي لتشومسكي (Chomsky hierarchy) وطبقات التعقيد (complexity classes) ضعيفة وليست تامة الوضوح .

رابعا : طبقتا التعقيد P و NP

The Complexity Classes P and NP

نرى أنه من المفيد الآن – على الأقل من الناحية التعليمية – أن نلخص فيما يلي الصعوبات التي واجهناها عند محاولتنا إيجاد طبقات تعقيد مفيدة (useful complexity classes) لِلُّغات الشكلية ، ونعرض بعض الاستنتاجات .

(١) يوجد عدد لا نهائي من طبقات التعقيد المتداخلة فعليًّا (properly nested complexity classes) $DTIME(n^k), k = 1, 2, \ldots$ وطبقات التعقيد هذه لها صلة بسيطة مع التدرج الهرمي المعتاد لتشومسكي ، ويبدو أنه من الصعب الوصول إلى نظرة عميقة في طبيعة هذه الطبقات . وقد لا تكون هذه طريقة جيدة لتصنيف اللغات .

(٢) النموذج الخاص بآلة تيورنج يؤثّر على درجة التعقيد ، حتى لو قَيَّدنا أنفسنا باستخدام الآلات المحددة (deterministic machines) . وليس من الواضح ما هو نوع آلة تيورنج الذي يُعَدُّ أفضل نموذج لحاسوب فعلي (actual computer) . ولذلك فيجب ألا يعتمد أي تحليل (analysis) على أي نوع خاص من آلات تيورنج .

(٣) وجدنا عدة لغات يمكن حسمها بكفاءة بآلة تيورنج غير محددة . ولبعض اللغات توجد أيضا خوارزميات محددة معقولة ، ولكن للبعض الآخر نعلم فقط طرقا قسرية منخفضة الكفاءة (inefficient bruteforce methods) . فما هي دلالات هذه الأمثلة ، وماذا تقتضي ؟

نظـرا لأن محـاولة إنتاج تدرجات هرمية لِلُّـغات وتكون ذوات معنى (meaningful language hierarchies) عـن طـريـق درجـات التعقيـد الزمنيـة (via time – complexities) مع معدلات نمو مختلفة (with different growth rates) تبدو محـاولةً غير منتجـة (unproductive) ، فلذلك سنتجاهل (ignore) بعض العوامل الأقل أهميـة ، مـثلا بإزالـة (removing) بعـض الاختـلافـات غـير الهـامـة (uninteresting distinctions) ، كتلك التي بين $DTIME\ (n^k)$ و $DTIME\ (n^{k+1})$. ويمكننا القول بـأن الاختـلاف مثـلا بيـن $DTIME\ (n)$ و $DTIME\ (n^2)$ ليـس جوهـريا / أسـاسيا (fundamental) ، نظرا لأن بعضه يعتمد على نموذج آلة تيورنج الخاص الذي نستخدمه [مثلا كم عدد الشرايط (tapes)] . وهذا يؤدي بنا إلى أن نأخذ في الاعتبار طبقة التعقيد المشهورة (famous complexity class)

$$P = \bigcup_{i \geq 1} DTIME(n^i)$$

هـذه الطبقـة تشمل جميع اللغات التي تقبلها آلة تيورنج محددة ما (some deterministic Turing machine) في زمن حدودية (in polynomial time) ، دون أي اعتبار لدرجة الحدودية . وكما رأينا سابقا فإن L_{REG} و L_{CF} تنتميان إلى P .

ونظـرا لأن التمييـز بيـن طبقـات التعقيـد المحـددة وغـير المحـددة (deterministic and nondeterministic complexity classes) يبدو جوهريا / أساسيا (fundamental) ، فلذلك نُعَرِّف أيضا

$$\mathrm{NP} = \bigcup_{i \geq 1} NTIME(n^i)$$

ومن الواضح أن

$$\mathrm{P} \subseteq \mathrm{NP}$$

ولكن ليس معلوما ما إذا كان هذا الاحتواء (containment) فعليا (proper) [أي : $\subset$] أم لا . وبينما يُعتقد عموما أن هناك بعض اللغات الموجودة في **NP** والتي ليست في **P** ، إلا أن أَحَدًا لم يجد للآن مثالا لذلك .

ويرجع الاهتمام بطبقات التعقيد هذه – وخاصة بالطبقة **P** – إلى محاولة التمييز بين العمليات الحاسوبية الواقعية وغير الواقعية (realistic and unrealistic computations). فهناك عمليات حاسوبية معينة رغم أنها ممكنة من الناحية النظرية ، إلا أن لها متطلبات موارد عالية تجعلها من الناحية العملية مرفوضة باعتبارها غير واقعية على الحواسيب الموجودة حاليا ، وأيضا على الحواسيب القوية / الممتازة / الضخمة / الكبيرة (supercomputers) التي سيتم تصميمها . مثل هذه المسائل يطلق عليها أحيانا " مسائل صعبة / عسيرة المعالجة " **(intractable problems)** للدلالة على أنها – رغم أنها من حيث المبدأ قابلة للحوسبة (computable) – لا يوجد لها أمل واقعي (realistic hope) في خوارزمية عملية (a practical algorithm) . ولفهم هذا بصورة أفضل حاول علماء الحاسوب وضع فكرة "صعوبة/ عُسر المعالجة " (intractability) في إطار / أساس شَكْلِي (formal basis) . وإحدى المحاولات لتعريف المصطلح " صعب / عَسِر المعالجة " **(intractable)** قد ظهرت فيما يطلُق عليها عموما " رسالة كوك – كارب " **(Cook – Karp thesis)** . وفي هذه الرسالة : أي مسألة تكون في **P** يطلق عليها " طَيِّعة / سهلة المعالجة " **(tractable)** ، وأي مسألة لا تكون في **P** يطلَق عليها " صعبة / عَسِرة المعالجة " **(intractable)** .

هل تُعَدُّ رسالة " كوك – كارب " طريقة جيدة لفصل المسائل التي نستطيع حلها في الواقع العملي عن تلك التي لا نستطيع حلها ؟ الإجابة ليست قاطعة . فمن الواضح أن أي عملية حاسوبية ليست في **P** تتزايد درجة تعقيدها الزمنية بمعدل أسرع من أي حدودية (grows faster than any polynomial) ، وتتزايد متطلباتها بسرعة كبيرة مع حجم المسألة (problem size) . وحتى بالنسبة لدالة مثل $2^{0.1\,n}$ ، فستكون الزيادة مفرطة لقيم n الكبيرة ، مثل $n \geq 1000$ ، وبالتالي قد نشعر أننا مُحِقُّون في تسمية مسألة بدرجة التعقيد هذه " صعبة المعالجة " . ولكن ماذا عن المسائل الموجودة في $DTIME\ (n^{100})$ ؟ بينما تطبِق رسالة " كوك – كارب " على مثل هذه المسألة " طَيِّعة / سهلة المعالجة " (tractable) لا نستطيع بالتأكيد أن نفعل الكثير معها ، حتى لقيم n الصغيرة . ويبدو أن المبرر لرسالة " كوك – كارب " يكمن في الملاحظة العملية التجريبية (empirical observation) أن معظم المسائل العملية (practical problems) الموجودة في **P** موجودة في $DTIME\ (n)$ أو $DTIME\ (n^2)$ أو $DTIME\ (n^3)$ ، بينما تلك الموجودة خارج هذه الطبقة تتجه درجات

تعقيدها إلى أن تكون أُسِّيَّة (exponential complexities) . فإذا نظرنا إلى المسائل العملية (practical problems) نجد أن هناك تمييزا واضحا (a clear distinction) بين المسائل التي تنتمي إلى **P** ، وتلك التي لا تنتمي إلى **P** .

خامسا : بعض مسائل NP — Some NP Problems

درس علماء الحاسوب كثيرا من مسائل **NP** ، أي المسائل التي يمكن حلها بطريقة غير مُحَدَّدة (**N**ondeterministically) في فترة زمنية حدودية (in **P**olynomial time) . وبعض الحجج والبراهين في تلك الدراسة تحتوي على تفاصيل فنية وأوصاف تقنية .

وعادةً تتم دراسة مسائل التعقيد (complexity questions) باعتبارها لغات (as languages) ، بحيث أن الحالات (cases) التي تُحَقِّق الشروط المقررة توُصف (described) بسلاسل في لغةٍ ما L (strings in same language) ، بينما تلك التي لا تحقق هذه الشروط تكون في $\overline{L}$ (مكمِّلة اللغة L) . ولذلك فعادةً أول شئ نحتاج إلى القيام به هو إعادة صياغة فهمنا البديهي للمسألة بدلالة لغةٍ ما .

مثال ٦ – ٦:

نعود هنا إلى النظر في مسألة التَّحَقُّقِيَّة SAT (satisfiability problem) التي عرضناها في مثال ٦ – ٢ . وكنا قد زعمنا بعد مناقشة أولية تمهيدية أن هذه المسألة يمكن أن تُحل بكفاءة عالية بآلة تيورنج غير محددة (nondeterministic Turing machine) ، وبكفاءة أقل أو بدون كفاءة نوعًا ما (rather inefficiently) ببحثٍ قَسْرِيٍّ أُسِّي (brute – force exponential search) . وكنا قد تجاهلنا (ignored) عددا من النقاط الفرعية / الثانوية (minor points) في حُجَّتِنا (argument) للوصول إلى تلك النتيجة .

نفرض أن تعبيرًا ما في الصيغة القياسية / المعيارية العَطْفِيَّة (Conjunctive Normal Form) **CNF**(expression) ، وفيه طوله n (length) حَرْفِيَّات مختلفة (different literals) عددها m . نظرا لأنه من الواضح أن $m < n$ يمكننا اعتبار n حجم المسألة (problem size) . وبعد ذلك يجب أن نُشَفِّر (encode) التعبير **CNF** كسلسلة لآلة تيورنج (as a string for a Turing machine) . ويمكننا إجراء ذلك مثلا بأخذ $\Sigma = \{x, \vee, \wedge, (,), -, 0, 1\}$ ، وبتشفير (encoding)

مؤشر x(subscript of) كعدد ثنائي في هذا النظام التعبير **CNF** $(x_1 \vee \overline{x_2}) \wedge (x_3 \vee x_4)$ يُشَفَّر هكذا :

$$(x1 \vee x - 10) \wedge (x11 \vee x100)$$

ونظرًا لأن المؤشر (subscript) لا يمكن أن يكون أكبر من m ، فإن أكبر طول (the maximum length) لأي مؤشر هو $\log_2 m$. وكنتيجة تتبع ذلك فإن أكبر طول مُشَفَّر (maximum encoded length) لأي صيغة قياسية عَطْفِيَّة **CNF** مكونة من n رمز (n – symbol CNF) هو $O(n \log n)$.

والخطوة التالية هي أن نُوَلِّد (generate) حلا تجريبيا [حَلَّ محاولةٍ (a trial solution)] للمتغيرات . وبطريقة عدم التحديد (nondeterministically) فإن هذا يمكن أن يتم في وقت time $O(n)$. وهذا الحل المحاولة يُعَوَّض (substituted) عندئذ في سلسلة الإدخال (input string) . وهذا يمكن إجراؤه في وقت time $O(n^2 \log n)$ [(*)] . وبالتالي فإن العملية الكُلِّيَّة (entire process) يمكن أن تُنَفَّذ في وقت $O(n^2 \log n)$ أو $O(n^3)$ ، والمسألة SAT تنتمي إلى المسائل غير المحدَّدة $SAT \in \mathbf{NP}$.

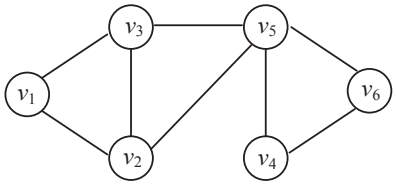
* * *

وهناك عدد كبير من مسائل المخططات البيانية (graph problems) التي تمت دراستها ، والمعلوم أنها تنتمي إلى **NP** .

مثال ٦ – ٧ : مسألة المسار " الهاملتوني " **The Hamiltonian Path Problem**

إذا أُعطينا مخططا بيانيا غير مُوَجَّه (an undirected graph) رؤوسه (vertices) $v_1, v_2, \ldots, v_n$ ، فإن المسار " الهاملتوني " هو مسار بسيط (a simple path) يمر بجميع الرؤوس . فالمخطط البياني الذي يظهر في شكل ٦ – ٢ يحتوي على المسار " الهاملتوني " $(v_2, v_1), (v_1, v_3), (v_3, v_5), (v_5, v_4), (v_4, v_6)$. ومسألة المسار

(*) هذا التقدير (estimate) يعتمد على كيفية رؤيتنا أو نظرتنا (visualizing) لهذه العملية (process) وإمكانية تحسينها (improving) . ولكن هذا الأمر ليس له أهمية نظرا لأننا لا نميِّز بين درجات الحدوديات (degrees of polynomials) .



شكل ٦ – ٢

" الهاملتوني " **(Hamiltonian path problem) (HAMPATH)** هي أن نُحَدِّد (decide) ما إذا كان مخطط بياني معطَى (a given graph) يحتوي على مسار " هاملتوني " أم لا .

من السهل أن نجد خوارزمية محدَّدة (a deterministic algorithm) ، وذلك لأن أي مسار " هاملتوني " هو تبديل (a permutation) للرؤوس $v_1, v_2, \ldots, v_n$. وهناك عدد $n!$ من هذه التباديل / التبديلات ، وإجراء بحث قسري (a brute – force search) لجميعها سيعطِي الإجابة . ولكن للأسف هذا تكلفته باهظة (a great expense) حتى لقيم n المتواضعة .

ولاستكشاف الحل غير المحدد (nondeterministic solution) يجب أن نجد أولا طريقة لتمثيل أي مخطط بياني (a graph) بسلسلة (a string) . وإحدى أبسط وأنسب (simplest and most convenient) الطرق لتشفير (encoding) المخططات البيانية هي باستخدام/ بتطبيق " مصفوفة تَجاور " **(an adjacency matrix)** .

تعريف ٦ – ٣ :

نفرض أن لدينا مخططا بيانيا مُوَجَّها (a directed graph) رؤوسه $v_1, v_2, \ldots, v_n$ ومجموعة أحرفه E (edge set) . تُعَرَّف " مصفوفة تَجَاوُر " **(an adjacency matrix)** لهذا المخطط بأنها منظومة (array) $n \times n$ ، العنصر $a(i,j)$ فيها (وهو العنصر الموجود في الصف i والعمود j) يحقق الشرط التالي :

$$a(i, j) = 1 \text{ if } (v_i, v_j) \in E,$$
$$= 0 \text{ if } (v_i, v_j) \notin E.$$

وأي حرف غير موجَّه (an undirected edge) يمكن أن نعتبره حرفين منفصلين (two separate edges) في اتجاهين متعاكسين (in opposite directions) .

مثال ٦ – ٨ :

المنظومة التالية تُمَثِّل المخطط البياني الذي يظهر في شكل ٦ – ٢ .

0 1 1 0 0 0
1 0 1 0 1 0
1 1 0 0 1 0
0 0 0 0 1 1
0 1 1 1 0 1
0 0 0 1 1 0

وهكذا نرى أن أي مخطط بياني يحتوي على عدد n من الرؤوس يتطلب تمثيله سلسلة طولها n^2 (a string of length) . وفي حالة المخطط غير الموجَّه فإن المصفوفة تكون متماثلة ، وبالتالي يمكن خفض (reducing) متطلبات التخزين (storage requirements) إلى $(n + 1)n/2$. ولكن في أي حالة سيكون طول سلسلة الإدخال (input string) هو $O(n^2)$.

وبعد ذلك نقوم بطريقة غير محددة (nondeterministically) بتوليد (generating) تبديل للرؤوس (a permutation of the vertices) . وهذا يمكن إجراؤه في وقت $O(n^3)$. وأخيرا نختبر التبديل (check the permutation) لنرى إن كان يُكوِّن مسارا (constitutes a path) أم لا . ويكفي لهذا وقت $O(n^4)$. ولذلك فإن $HAMPATH \in NP$.

مثال ٦ – ٩ :

مسألة العُصْبَة The Clique Problem

نفرض أن G مخطط بياني غير موجَّه (an undirected graph) رؤوسه (vertices) هي $v_1, v_2, \ldots, v_n$. العصبة k- **clique**) k- هي مجموعة جزئية (subset) $V_k \subseteq 2^V$ ، بحيث أنه يوجد حرف (an edge) بين رأسي أي زوج من أزواج الرؤوس $v_i, v_j, \in V_k$ (every pair of vertices) . ومسألة العُصْبة (**CLIQ**) (the clique problem) هي أن نحدد (decide) ما إذا كان المخطط G يحتوي على عُصْبة k- أم لا ، حيث k عدد صحيح معطَى .

ويمكن لبحث محدَّد (a deterministic search) أن يختبر (examine) جميع عناصر 2^V . وهذا إجراء مباشر (straightforward) ، ولكنه يتميز بدرجة تعقيد زمنية أُسِّيَّة (exponential time – complexity) . وأما أي خوارزمية غير محددة (a nondeterministic algorithm) فإنها تقوم بمجرد تخمين المجموعة الجزئية الصحيحة (just guesses the correct subset) . وتمثيل المخطط البياني والتَّحَقُّق (checking) شبيهان بما تَمَّ في المثال السابق . وبناءً عليه نزعم بدون إجراء أي خطوات أخرى أنه يمكن حل مسألة العُصْبة في وقت $O(n^4)$ ، وأن

$$CLIQ \in \text{NP}$$

* * *

هناك مسائل أخرى كثيرة كتلك التي عرضناها : بعضها مماثلة (similar) لمسائل الأمثلة السابقة ، والأخرى مختلفة عنها تماما ، ولكنها جميعا تشترك في الخصائص نفسها :

(١) جميع المسائل تنتمي إلى **NP** ، ولها حلول بسيطة غير محددة (simple nondeterministic solutions) .

(٢) جميع المسائل لها حلول محددة درجة تعقيدها الزمنية أُسِّيَّة (deterministic solutions with exponential time – complexity) ، ولكن ليس معلوما ما إذا كانت طَيِّعة / سهلة المعالجة (tractable) أم لا .

وللنظر بصورة أعمق في الصلة بين هذه المسائل المتنوعة نحتاج لإيجاد بعض العموميات المشتركة (some commonality) لجميع هذه الحالات التي تبدو مختلفة .

سادسا : الاختزال في فترة زمنية حدودية

Polynomial – Time Reduction

إحدى طرق توحيد (unifying) الحالات المختلفة هي أن نرى إن كان بإمكاننا اختزالها (reducing them) إلى بعضها البعض ، بمعنى / بمفهوم (in the sense) أنه إذا كانت إحداها طيِّعة / سهلة المعالجة فستكون الأخريات أيضا سهلة المعالجة .

تعريف ٦ – ٤ :

يقال لِلُغَةٍ L_1 إنها قابلة للاختزال في فترة زمنية حدودية (polynomial – time reducible) إلى لغةٍ أخرى L_2 إذا وُجدت آلة تيورنج محدَّدة يمكننا بها تحويل (transforming) أي سلسلة w_1 في المجموعة الأبجدية (alphabet) للغة L_1 إلى سلسلة w_2 في المجموعة الأبجدية للغة L_2 في فترة زمنية حدودية (in polynomial time) بطريقة ما بحيث أن $w_1 \in L_1$ إذا وفقط إذا كانت $w_2 \in L_2$.

مثال ٦ – ١٠ :

في مسألة التحقُّقِيَّة (satisfiability problem) لم نضع أي قيد (restriction) على طول أي عبارة (length of a clause) . أحد الأنواع المقيَّدة (a restricted type) من مسائل التحقُّقِيَّة مسألة التَّحَقُّقِيَّة الثلاثية (3SAT) (three – satisfiability problem) والتي يمكن فيها لأي عبارة (clause) أن تحتوي على الأكثر (at most) على ثلاثة حَرْفِيَّات (three literals) . ومسألة SAT مسألة قابلة للاختزال في فترة زمنية حدودية إلى 3SAT (polynomial – time reducible to) .

ونوضح عملية الاختزال (reduction) بالتعبير البسيط رباعي – الحَرْفِيَّات (4-titeral expression)

$$e_1 = (x_1 \vee x_2 \vee x_3 \vee x_4)$$

نُعَرِّف متغيرا جديدا z ، وننشئ التعبير

$$e_2 = (x_1 \vee x_2 \vee z) \wedge (x_3 \vee x_4 \vee \bar{z})$$

إذا كان e_1 صحيحا / صادقا (true) ، فإن أحد الحَرْفِيَّات x_1, x_2, x_3, x_4 يجب أن يكون صحيحا (true) . فإن كان $x_1 \vee x_2$ صحيحا (true) ، فإننا نختار $z = 0$ ، ويصبح e_2 صحيحا (true) . وإن كان $x_3 \vee x_4 = 1$ ، فيمكننا اختيار $z = 1$ لتحقيق e_2 (أي ليكون e_2 صحيحا) . وبالعكس إذا كان e_2 صحيحا ، فإن e_1 أيضا يجب أن يكون صحيحا . وبالتالي فبالنسبة للتحققية التعبيران e_1, e_2 متكافئان (equivalent) .

ونحن نزعم أن هذا النموذج (pattern) يمكن أن يُمَدَّد / يوسَّع (extended) إلى عبارات (clauses) تحتوي على أكثر من أربعة حَرْفِيَّات (four literals) ، ولكننا سنترك البرهان كتدريب للقارئ الكريم .

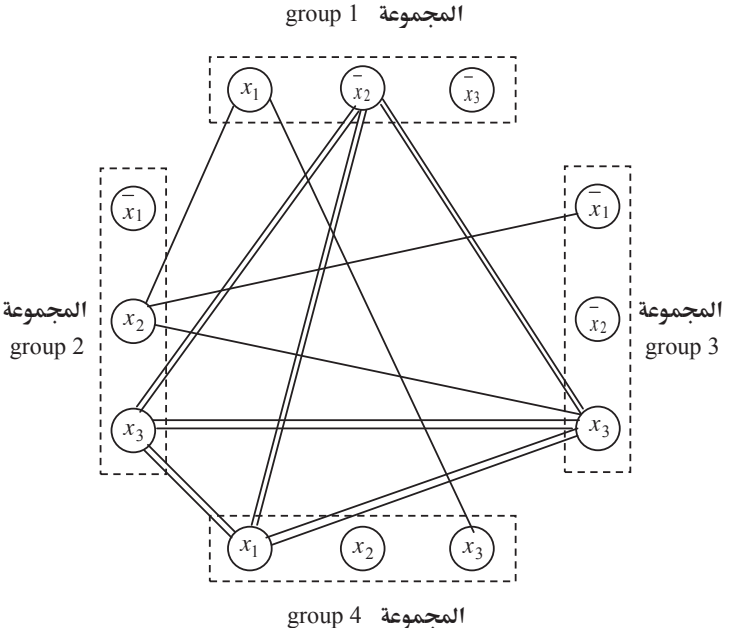
مثال ٦ – ١١ :

المسألة التحقُّقِيَّة الثلاثية **3SAT** قابلة للاختزال إلى مسألة العُصْبَة **CLIQUE** في فترة زمنية حدودية (in polynomial – time) .

يمكننا أن نفترض أنه في أي تعبير **3SAT** تحتوي أي عبارة (clause) على ثلاثة حَرْفِيَّات (three literals) بالضبط . وإذا لم يكن هذا هو الحال في بعض التعابير ، فإننا نقوم بمجرد إضافة حَرْفِيَّات إضافية (extra literals) لا تُغَيِّر التحقُّقِيَّة . فمثلا $(x_1 \vee x_2)$ يكافئ $(x_1 \vee x_1 \vee x_2)$. والآن نأخذ في الاعتبار التعبير

$$e=(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

نرسم مخططا بيانيا (a graph) تُمَثِّلُ فيه كلُّ عبارة (each clause) بمجموعة من ثلاثة رؤوس (a group of three vertices) ، ويرتبط فيه (associated) كلُّ حَرْفِيّ (each literal) بإحدى الرؤوس [انظر شكل ٦ – ٣] .



شكل ٦ – ٣

لكل رأسٍ (vertex) في مجموعةٍ نَضَعُ حرفا (put in an edge) لجميع رؤوس المجموعات الأخرى ، ما لم يكن الحَرْفِيَّان المرتبطان (the two associated literals) مُكَمِّلَيْن (complements) لبعضيهما . ففي شكل ٦ – ٣ نرسم حرفًا (*) من $(\overline{x_2})_1$ إلى $(x_3)_2$ ، ومن $(\overline{x_2})_1$ إلى $(x_3)_3$ ، ولكن ليس من $(\overline{x_2})_1$ إلى $(x_2)_2$. ويبيّن شكل ٦ –٣ مجموعة جزئية (subset) فقط من الأحرف (edges) ، حيث أن شكل المجموعة الكاملة سيبدو معقَّدا للغاية ومتشابكا . ولاحظ أن المخطط البياني الجزئي (subgraph) الذي رؤوسه $(\overline{x_2})_1$, $(x_3)_2$, $(x_3)_3$, $(x_1)_4$ هو عُصْبَة - 4 (4 – clique) ، وأن $\overline{x_2} = x_3 = x_1 = 1$ هو إسناد متغيرات (a variable assignment) يحقق التعبير ***e*** .

هذه الطريقة يمكن أن تُعَمَّمَ (generalized) لأي تعبير (expression) **3SAT** مع ***k*** من العبارات . ويمكن إثبات أن مسألة التحققية الثلاثية (3SAT problem) يمكن أن تتحقق (be satisfied) إذا وفقط إذا احتوى المخطط البياني المرافق (associated graph) على عُصْبَة - k (a k – clique) . وزيادةً على هذا ليس من الصعب أن نرى أن التحويل (transformation) من التعبير (expression) **3SAT** إلى المخطط البياني يمكن أن يتم إجراؤه بطريقة محدَّدة (can be done deterministically) في فترة زمنية حدودية (in polynomial time) .

* * *

وأهمية هذه الاختزالات تكمن في أننا نستطيع الآن النظر بعدة طرق إلى مسألةٍ معطاة . مثلا نفرض أننا نُخَمِّن أن مسألة **SAT** سهلة المعالجة (tractable) . فإن كان من الصعب إثبات هذا ، فقد نحاول مع الحالة الأبسط **3SAT** . وإذا لم نفلح مع هذه أيضا ، فيمكننا محاولة إيجاد خوارزمية ذات كفاءة عالية (an efficient algorithm) لمسألة العُصْبَة (the clique problem) ، أو لمسألة أخرى ذات علاقة من مسائل المخططات البيانية (some other related graph problem) . فإذا أمكننا إثبات أن أيًّا من هذه الخيارات سهل المعالجة (tractable) ، فيمكننا أن نزعم أن المسألة **SAT** كذلك سهلة المعالجة .

(*) المؤشِّر (subscript) الثاني يحدِّد رقم المجموعة (group) التي يَنتمي إليها الرأس (vertex) .

سابعا : التمام / الكمال – NP وسؤال قيد البحث
NP – Completeness and an Open Question

هناك عدد من المسائل التي تُعَدُّ أساسية ومركزية (central) بالنسبة لدراسة درجة التعقيد (complexity study) ، والتي إذا فهمنا تماما واحدة منها ، فسنفهم بإذن الله المحور الأساسي والموضوع الرئيسي الذي ترتكز وتشتمل عليه خاصية سهولة / يُسْر المعالجة (tractability) .

تعريف ٦ – ٥ :

يُقال لِلُغَةٍ إنها " تامة / كاملة – NP (NP – complete) إذا كانت $L \in NP$ ، وكانت أي لغة $L_1 \in NP$ قابلة للاختزال إلى اللغة L في فترة زمنية حدودية (polynomial– time reducible to L) .

ينتج من هذا التعريف أنه إذا كانت اللغة L كاملة - NP وقابلة للاختزال إلى اللغة L_1 في فترة زمنية حدودية ، فإن اللغة L_1 ستكون أيضا كاملة - NP . وهكذا فإذا استطعنا أن نجد خوارزمية مُحَدَّدَة واحدة ذات فترة زمنية حدودية (one deterministic polynomial – time algorithm) لأي لغة كاملة - NP ، فإن أي لغة في NP ستكون أيضا في P ، أي أن

$$P = NP$$

وهنا يصبح لدينا أمل في وجود خوارزميات ذوات كفاءة عالية (efficient algorithms) لمثل هذه المسائل ، رغم أننا لم نجد واحدةً حتى الآن . ومن ناحية أخرى إذا استطعنا إثبات أن أيًّا من " المسائل الكاملة - NP المعلومة العديدة " صعبة المعالجة (intractable) ، فإن كثيرا من المسائل الهامة ستكون غير قابلة للحل عمليا (not practically solvable) . وهذا يضع خاصية الكمال - NP في مركز موضوع سهولة المعالجة (tractability) لمسائل هامة كثيرة . وعند هذه النقطة نحتاج لدراسة بعض المسائل الكاملة - NP . والنتيجة التالية والتي تُعْرَف باسم " نظرية كوك " (**Cook's theorem**) تعتبر نقطة الدخول إلى هذه الدراسة .

نظرية ٦ – ٥ :

مسألة التَّحَقُّقِيَّة **SAT** (Satisfiability Problem) كاملة **- NP** .

البرهان :

فكـرة البرهـان هـي أنـه لكـل متتابعـة تشـكيلة آلـة تيـورنج (configuration sequence of a Turing machine) نستطيع إنشاء تعبير **CNF**(expression) يكون تَحَقُّقِيًّا (satisfiable) إذا وفقط إذا وُجِدَت متتابعة من التشكيلات (a sequence of configuration) مؤدية إلى القبول (leading to acceptance) . وتفاصيل البرهان طويلة ومعقدة وتلقى ضوءا بسيطا على مسألة **NP** ، ولذلك ستُحذف هنا ويمكن الرجوع إليها في الكتب المتخصصة في موضوع درجة التعقيد (complexity) .

* * *

وإذا قَبلنا نظرية " كوك " (Cook's theorem) فسيكون لدينا على الفور عدد من المسائل الكاملة **- NP** .

مثال ٦ – ١٢ :

أثبتنـا سابقـا أن المسـألة **SAT** يمكن أن تُختزل إلى المسألة **3SAT** ، وأن المسألة **3SAT** يمكن أن تُختزل إلى المسألة **CLIQ** . ولذلك فالمسألتان **3SAT, CLIQ** كلتاهما كاملتان **- NP** (**NP** – complete) .

وينـتج كـذلك أن المسـألة **HAMPATH** هـي أيضـا كاملـة – **NP** ، ولكـن الاختزالات التي نحتاجها لبيان ذلك تُعَد أقل وضوحا .

* * *

وبالإضافة إلى المسائل SAT, 3SA**T**, CLIQ, HAMPATH توجد مسائل أخرى كثيرة معلوم أنها كاملة – **NP** . وقد بُذِل جهد كبير في محاولة إيجاد خوارزميات ذوات كفاءة عالية لكل من هذه المسائل ولكن دون أن تُكلل بالنجاح حتى الآن . وهذا يؤدي بنا إلى التخمين بأن

$P \neq NP$

وأن مسائل هامة كثيرة صعبة المعالجة (intractable) . ولكن رغم أن هذا تخمين عملي معقول إلا أنه لم يتم إثباته . وتبقى هذه هي المسألة الأساسية المفتوحة للبحث (open problem) في نظرية التعقيد (complexity theory) .

تمرينات رقم (٦)

أولا : كفـاءة الحوسبـة

٦ – ١) راجع خوارزميات عملية الترتيب / الفَرْز (sorting) لترى كيف أن اختيار الخوارزمية يؤثّـر على كفاءة هذه العملية . ما هي درجة التعقيد الزمنية (time complexity) لأكفأ خوارزميات الفَرْز (most efficient sorting algorithms) ؟

ثانيا : نماذج آلات تيورنج ودرجة التعقيد

٦ – ٢) أعد كتابة التعبير البُولي (boolean expression)

$$(x_1 \wedge x_2) \vee x_3$$

بالصيغة القياسية العَطْفِيَّة **CNF** (conjunctive normal form) .

٦ – ٣) في برهان نظرية ٦ – ٢ أهملنا نقطة دقيقة ، وهي أنه عندما تكبر / تنمو (grows) تشكيلةٌ ما (a configuration) ، فإن بقية محتويات الشريط يجب أن يتم تحريكها . هل هذا الإهمال يؤثر على النتيجة التي وصلنا إليها ؟

ثالثا : عائلات اللغات وطبقات التعقيد

٦ – ٤) اثبت أنه توجد لغات لا تنتمي إلى $NTIME(2^n)$.

بعض مسائل NP

٦ – ٥) في مسألة **HAMPATH** وَضِّـح كيف أنه يمكن توليد تبديل (generating a permutation) بطريقة غير محددة (nondeterministically) في فترة زمنية $O(n^3)$ time .

الاختزال في فترة زمنية حدودية

٦-٦) " مسألة البائع المسافر " **TSP** (The Traveling Salesman Problem)
يمكن أن تُصاغ كما يلي : نفرض أن G مخطط بياني كامل / تام (a complete graph) أُسندت (assigned) لأحرفه (edges) أوزان غير سالبة (nonnegative weights) . وَوَزن (weight) أي مسار بسيط (a simple path) هو مجموع جميع أوزان الأحرف (edge weights) . ومسألة TSP هي أن نحسم (decide)/ نقرر – لأي عدد صحيح مُعْطَى $k \geq 0$ – ما إذا كان المخطط البياني G يحتوي على مسار " هاملتوني " (a Hamiltonian path) ذي وزن أقل من أو يساوي k .
اثبت أن المسألة **HAMPATH** قابلة للاختزال إلى المسألة **TSP** في فترة زمنية حدودية (HAMPATH is polynomial – time reducible to **TSP**) .

أجوبة تمرينات الكتاب

أجوبة تمرينات رقم (١)

١ – ١ – أ) الحل التالي يحتوي على ثلاث حالات ويغطي جميع المدخلات (scans the entire input) :

$$\delta(q_0, a) = (q_1, a, R),$$
$$\delta(q_1, a) = \delta(q_1, b) = (q_1, a, R),$$
$$\delta(q_1, \square) = (q_2, \square, R),$$
$$F = \{q_2\}.$$

ب) نعم يمكن تنفيذ التصميم بآلة ذات حالتين بمجرد فحص (examining) الرمز الأول وإهمال بقية المدخلات ، مثلا

$$\delta(q_0, a) = (q_2, a, R)$$

لاحظ أنه في آلة تيورنج ليس من الضروري فحص جميع المدخلات قبل قبولها .

١ – ٢ – أ)

$$\delta(q_0, a) = (q_1, a, R),$$
$$\delta(q_1, b) = (q_2, b, R),$$
$$\delta(q_2, a) = (q_2, a, R),$$
$$\delta(q_2, b) = (q_3, b, R),$$
$$F = \{q_3\}.$$

ب)

$$\delta(q_0, a) = (q_0, b) = (q_1, \square, R),$$
$$\delta(q_0, \square) = (q_2, \square, R),$$
$$\delta(q_1, a) = \delta(q_1, b) = (q_0, \square, R),$$
$$F = \{q_2\}.$$

١ – ٣) الحل بسيط في مفهومه ، ولكنه ممل وطويل في كتابة تفاصيله . والمخطط العام للحل يبدو شيئا كالتالي :

i) ضع رمزاً كعلامة مميزة c (a marker symbol) عند كل نهاية للسلسلة .

ii) استبدل بكل توافق (combination) من الرمزين ca على اليسار التوافق ac ، وبكل توافق من الرمزين ac على اليمين التوافق ca .
كرر ذلك حتى يلتقي الرمزان cc في منتصف السلسلة .

iii) احذف أحد الرمزين cc ، وحرِّك بقية السلسلة لملء الفجوة (gap) .
من الواضح أن هذه مهمة طويلة ، ولكنها نمطية بالنسبة للطرق الطويلة والمملة التي تتبعها عادة آلات تيورنج لتنفيذ عمليات وأشياء بسيطة .

١ – ٤) لا نستطيع مجرد البحث في اتجاه واحد نظرا لأننا لا نعلم متى نتوقف . وإنما يجب علينا أن نتحرك في الاتجاهين يمينا ويسارا جيئة وذهابا ، ونضع علامتين مميزتين (markers) يمينا ويسارا عند حدود (boundaries) المنطقة التي تم البحث فيها (searched region) ، مع تحريك هاتين العلامتين للخارج (outward) .

١ – ٥) نعم صحيح .. إذا كانت مجموعة الحالات النهائية F تحتوي على أكثر من عنصر واحد ، عَرِّف حالة نهائية جديدة q_f والانتقالات

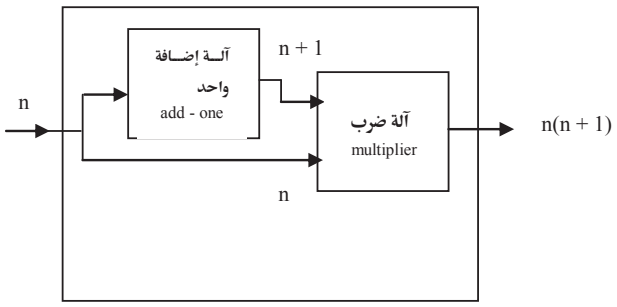
$$\delta(q, a) = (q_f, a, R)$$

$$\forall\ q \in F,\ a \in \Gamma\ .$$

١ – ٦) يمكننا أن ننظر للآلة على أنها مكونة من جزئين أساسيين :

- آلة إضافة واحد (**add – one** machine) تقوم بمجرد إضافة واحد 1 إلى (جَمْع واحد 1 على) المدخلات (input) .
- وآلة ضرب (**multiplier**) تقوم بضرب عددين .

ونجمع بين الآلتين في المخطط القالبي البسيط التالي :



١ – ٧) أولاً نَفْصِل (split) المدخلات إلى جزئين متساويين ، ويمكننا تنفيذ ذلك بالطريقة التي اقترحناها في حل السؤال ١ – ٣) . ثم نقارن الجزئين ، كل رمز مع الرمز المقابل إلى أن نجد عدم توافق (a mismatch) .

١ – ٨) أحد الحلول :

$$\delta(q_0, a) = (q_i, a, R),$$
$$\delta(q_0, c) = (q_0, c, R) \quad \forall \quad c \in \Sigma - \{a\},$$
$$\delta(q_0, \square) = (q_j, \square, R).$$

الحالة q_0 هي أي حالة يمكن أن تُطبَّق عندها التعليمة searchright .

١ – ٩) تَجَاهَلْنا حقيقة أن آلة تيورنج – كما هي مُعَرَّفة حتى الآن – آلة مُحَدَّدَة (deterministic) ، بينما آلة الدفع لأسفل pda يمكن أن تكون غير مُحَدَّدَة (nondeterministic) . ولذلك لا يمكننا حتى الآن أن نزعم أن آلات تيورنج أقوى وأكثر قدرة من آلات الدفع لأسفل .

أجوبة تمرينات رقم (٢)

٢ – ١ – أ) الآلة لها دالة انتقال

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$$

مع وجود القيد أنه لجميع الانتقالات $\delta(q_i, a) = (q_j, b, L \text{ or } R)$ فإن الشرط $a = b$ يجب أن يتحقق .

ب) لمحاكاة $\delta(q_i, a) = (q_j, b, L)$ حيث $a \neq b$ في الآلة القياسية ، نُعَرِّف انتقالات جديدة $\delta(q_i, a) = (q_{jL}, b, S)$ و $\delta(q_{jL}, b) = (q_j, b, L)$ ، وذلك لجميع العناصر $c \in \Gamma$ ، وهكذا .

٢ – ٢) نُعَرِّف فراغا زائفا **B** (a pseudo-blank) . وكلما أرادت الآلة الأصلية كتابة □ ، فإن الآلة الجديدة تكتب **B** . وبعد ذلك لأي انتقال $\delta(q_i, \square) = (q_j, b, L)$ فإننا نضيف الانتقال $\delta(q_i, B) = (q_j, b, L)$ ، وهكذا . وبالطبع فإننا يجب أن نحتفظ بالانتقال الأصلي $\delta(q_i, \square) = (q_j, b, L)$ لمعالجة الفراغات الموجودة أصلا على الشريط .

٢ – ٣) لا يحد هذا القيد من قدرة الآلة . فلكل رمز $a \in \Gamma$ نُعَرِّف رمزا زائفا (a pseudo-symbol) ، وليكن A . وكلما احتجنا إلى الاحتفاظ (preserving) بهذا الرمز a ، فإننا نكتب أولا A ، ثم نعود إلى الخلية المعنية لنستبدل بالرمز A الرمز a .

٢ – ٤) نستبدل بالانتقال

$$\delta (q_i, \{a, b\}) = (q_j, c, R)$$

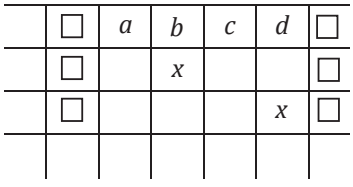
الانتقال

$$\delta (q_i, d) = (q_j, c, R)$$

وذلك لجميع العناصر $d \in \Gamma - \{a, b\}$

٢ – ٥) بالنسبة للتعريف الشكلي نستخدم $\Gamma \times \Gamma \times \ldots \times \Gamma = \Gamma_T$ وكذلك $\delta : Q \times \Gamma_T \rightarrow Q \times \Gamma_T \times \{L, R\}^m$ ، حيث m هي عدد رؤوس القراءة والكتابة . وأحد الأمور التي يجب أخذها في الاعتبار هو : ماذا يحدث عندما تكون هناك رأسان للقراءة والكتابة واقفتان على الخلية نفسها ؟

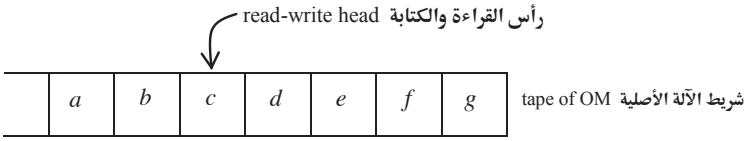
التعريف الشكلي يجب أن يقدِّم حلا للتعارضات (conflicts) الممكنة . لمحاكاة الآلة الأصلية **OM** (Original Machine) بآلة تيورنج قياسية (Standard Machine) **SM** ، فإننا نجعل الآلة القياسية **SM** تحتوي على عدد $m + 1$ من المسارات (tracks) . وعلى أحد هذه المسارات نحتفظ بمحتويات شريط الآلة الأصلية **OM** ، بينما نستخدم المسارات الأخرى – وعددها m – لبيان مواضع رؤوس شريط الآلة الأصلية **OM** (**OM's** tape heads) .

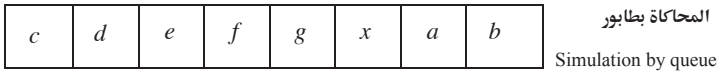


الآلة القياسية **SM** ستحاكي كل حركة من حركات الآلة الأصلية **OM** عن طريق مسح (scanning) وتحديث (updating) منطقتها الفَعَّالة (its active area) .

٢ – ٦) هذا السؤال يبيِّن أن أي آلة طابور تكافئ آلة تيورنج قياسية ، وبالتالي فإن الطابور يُعَدُّ أداة تخزين أقدر (more powerful) من الرصة . ولمحاكاة آلة تيورنج قياسية (standard) **TM** بآلة طابور (a queue machine) ، يمكننا مثلا أن نحتفظ بالجانب الأيمن من الآلة الأصلية **OM** (right side of) في مقدمة الطابور (front of the queue) ، وبالجانب الأيسر في مؤخرته (back) ، كما يوضح ذلك الشكل التالي .

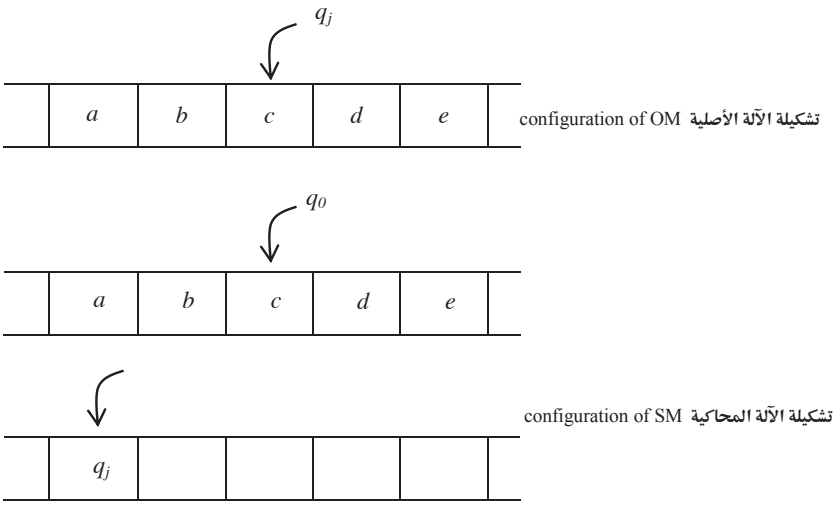
الحركة لليمين تعد سهلة ، حيث أننا نقوم بمجرد حذف (removing) الرمز الموجود في مقدمة الطابور (front symbol in the queue) ونضع شيئا ما في المؤخرة .





وأما الحركة لليسار فإنها تتجه عكس المزاج / الميل الفطري (against the grain) ، ولذا فيجب أن ندير (circulate) محتويات الطابور عدة مرات حتى نصل بجميع الرموز إلى أماكنها الصحيحة . ويساعدنا هنا أن نستخدم علامتين إضافيتين .

٢ – ٧) نحتاج لشريطين فقط : أحدهما يكون مرآة لشريط (mirrors the tape) الآلة الأصلية **OM** ، والآخر يخزِّن (stores) حالة (state) الآلة الأصلية **OM** .



الآلة المحاكية **SM** تحتاج إلى حالتين فقط : حالة قابلة (an accepting state) ، وحالة غير قابلة (a nonaccepting state) .

٢ – ٨) (i) ابدأ عند يسار المدخلات (left of the input) . تذكَّر الرمز بوضع الآلة في الحالة المناسبة (the appropriate state) . ثم استبدل بها الرمز X .

(ii) حرِّك رأس القراءة والكتابة إلى اليمين ، وتوقَّف (stopping) [بطريقة غير محدَّدة (nondeterministically)] عند مركز المدخلات (center of the input) .

(iii) قارن الرمز (symbol) الموجود هناك بالرمز الذي نتذكَّره (remembered one) . فإذا تواءما (match) ، فاكتب الرمز Y في الخلية . وإذا لم يتواءما فإننا نرفض (reject) المدخلات .

(iv) وبوجود العلامة Y (mark) عند مركز المدخلات ، فيمكننا الآن التقدُّم بطريقة محدَّدة (proceed deterministically) ، متحركين تباديا (alternatively) لليسار ولليمين ، مقارنين الرموز (comparing symbols) .

وللوصول إلى حل محدَّد كاملا (a completely deterministic solution) فإننا نوجد أولا مركز المدخلات [مثلا بوضع علامات (markers) عند كل نهاية ، وتحريكها للداخل (inwards) حتى تلتقي] .

٢ – ٩) اختر قيمة للعدد الصحيح n . وللوصول إلى ذلك قم بتوليد (generating) 1, 2, 3, … متوقفاً (stopping) بطريقة غير محدَّدة (nondeterministically) عند قيمةٍ ما n . ثم حدِّد ما إذا كان طول المدخلات (length of the input) هو أحد مضاعفات العدد الصحيح n (a multiple of) أم لا . فإن كان كذلك ، فاقبل (accept) . وإذا كان $a^n \in L$ ، فمعنى ذلك أنه يوجد عدد صحيح ما n يتحقق عنده ذلك .

٢ – ١٠) إحدى الرصتين ستحتفظ بمحتويات الشريط الموجودة يمين نقطة إسنادٍ ما (some reference point) ، بينما تحتفظ الرصة الأخرى بمحتويات الشريط الموجودة يسار النقطة . ثم يتم تنفيذ التحركات (moves) لليسار ولليمين ببساطة عن طريق الرَّفْع (popping) من الرصة والدَّفع (pushing) في الرصة .

٢ – ١١) فيما يلي مخطط لخوارزمية تؤدي المطلوب :

(i) ابدأ بنسخة من السلسلة السابقة (a copy of the preceding string) .

(ii) أوجد الصفر 0 الذي موضعه أقصى اليمين (rightmost 0) . غيِّره إلى واحد 1 . ثم غَيِّر جميع الآحاد 1's الموجودة يمينه إلى أصفار 0's .

(iii) إذا لم يكن هناك أي أصفار ، فغيِّر جميع الآحاد 1's إلى أصفار 0's ، وأضف واحدا 1 على اليسار .

(iv) كرر الخطوات السابقة ابتداءً من الخطوة i) .

٢ – ١٢) نفرض أن $S_1 = \{s_1, s_2, \ldots\}$, $S_2 = \{t_1, t_2, \ldots\}$. اتحاد (union) هاتين المجموعتين يمكن عَدُّه / تعديده (be enumerated) بكتابة

$$S_1 \cup S_2 = \{ s_1, t_1, s_2, t_2, \ldots\}$$

وإذا تطابق عنصران فإننا نكتبهما في قائمة العناصر مرة واحدة فقط . وهكذا نرى أن اتحاد المجموعتين سيكون أيضا قابلا للعد . وبالنسبة للمجموعة $S_1 \times S_2$ استخدم الترتيب (ordering) المبين في شكل ٢ – ١٧ .

٢ – ١٣) أولا نَقْسِم المدخلات على اثنين (divide the input by two) ، ونُحرِّك النتيجة إلى جزء واحد من الشريط (one part of tape) . ثم يمكننا بعد ذلك استخدام الحيِّز الذي أصبح فارغا (free space) – والذي كانت تَشْغَلُه المدخلات ابتداءً (initially occupied by input) – في تخزين القواسم المتتابعة (storing successive divisors) .

٢ – ١٤) استخدم آلة ذات مسارات ثلاثة كالمبينة في الشكل التالي . على المسار الثالث نحتفظ بقيمة المحاولة الحالية لـ $|w|$. وعلى المسار الثاني نضع مُقَسِّمات (dividers) متساوية المسافات فيما بينها ، حيث المسافة بين كل مقسِّمين تساوي عدد $|w|$ من الخلايا (cells) . ثم نقارن بين محتويات الخلايا الموجودة بين العلامات (markers) .

	a	*b*	*c*	*d*	*b*	*c*	*d*	
			x			x		
	1	1	1					

المدخلات input
المقسِّمات dividers
trial value of $|w|$
قيمة المحاولة لـ $|w|$

أجوبة تمرينات رقم (٣)

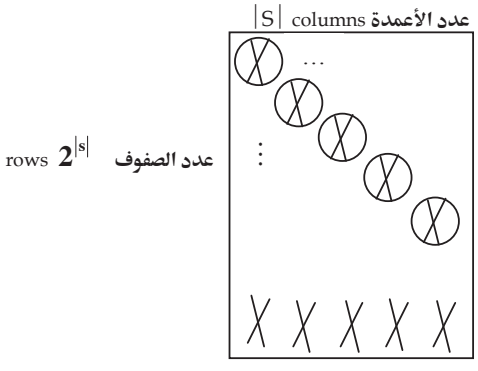
٣ – ١) نعلم أن اتحاد (union) مجموعتين قابلتين للعد هو مجموعة قابلة للعد (countable) . فإذا كانت مجموعة جميع اللغات غير القابلة للتعديد ارتداديا قابلة للعد أيضا ، فمعنى ذلك أن مجموعة جميع اللغات ستكون قابلة للعد . ولكننا نعلم أن هذا غير صحيح .

٣ – ٢) نفرض أن L_1, L_2 لغتان قابلتان للتعديد ارتداديا ، وأن M_1, M_2 هما آلتا تيورنج المقابلتان اللتان تقبلان هاتين اللغتين . وعندما يتم التمثيل بسلسلة مدخلات w (represented with an input string) ، فإننا نختار بطريقة غير محدَّدة (nondeterministically) الآلة M_1 ، أو الآلة M_2 لتشغيل / لمعالجة w (to process) . والنتيجة هي آلة تيورنج تقبل $L_1 \cup L_2$.

٣ – ٣) أي لغة حرة السياق هي لغة ارتدادية . وبالتالي فبنظرية ٣ – ٤ مكمِّلتها أيضا لغة ارتدادية . ولكن لاحظ أن اللغة المكمِّلة ليست بالضرورة حرة السياق .

٣ – ٤) لأي سلسلة معطاة w ، حيث $w \in L^+$ ، نأخذ في الاعتبار جميع التقسيمات $w = w_1 w_2 \ldots w_m$ (all splits) . لكل تقسيمة نحدِّد ما إذا كانت $w_i \in L$ أم لا . ونظرا لأنه لأي سلسلة w يوجد عدد محدود فقط من التقسيمات ، فلذلك يمكننا تحديد ما إذا كانت $w \in L^+$ أم لا .

٣ – ٥) البرهان الذي يحاول أن يثبت بالإقطار (diagonalization) أن 2^S ليست قابلة للعد لمجموعة محدودة S لا يصلح نظرا لأن الجدول المبين في شكل ٣ – ٢ ليس مربعاً ، حيث عدد الصفوف يساوي $|2^S|$ ، بينما عدد الأعمدة يساوي $|S|$.



عندما نقوم بعملية الإقطار ، فإن النتيجة على القطر يمكن أن تكون في أحد الصفوف السفلي .

٣ – ٦) ننظر أولا إلى اشتقاق نمطي (a typical derivation) :

$$S \overset{*}{\Rightarrow} aS_1bB \Rightarrow aaS_1bbB \overset{*}{\Rightarrow} a^nS_1b^nB \Rightarrow a^{n+1}b^{n-1}B \Rightarrow a^{n+1}b^{n+1}B \Rightarrow \ldots$$

ومن هذا الاشتقاق يمكننا أن نخمن أن القاعدة تولِّد اللغة

$$L = \{ a^{n+1}b^{n+k}, n \geq 1, k = -1, 1, 3, \ldots\}$$

٣ – ٧) رسميا / شكليا يمكننا وصف القاعدة هكذا $G = (V, S, T, P)$
حيث $S \subseteq (V \cup T)^+$
و

$$L(G) = \{x \in T^* : s \Rightarrow_G x \text{ for any } s \in S\}$$

والقواعد غير المقيَّدة المُعَرَّفة في تعريف ٣ – ٣ تُعَدُّ مكافئة لهذه القاعدة ذات الامتداد (extension) نظراً لأنه إذا أُعطينا أي قاعدة غير مقيَّدة فإنه يمكننا دائما إضافة قواعد ابتداء (starting rules) : $S_0 \to s_i \quad \forall\, s_i \in S$.

٣ – ٨) للوصول إلى هذه الصيغة المطلوبة للقواعد غير المقيَّدة ، نُعَرِّف / نُدْخِل متغيرات شكلية (dummy variables) على الطرف الأيمن كلما كان $|\, v\, | > |\, u\, |$. فمثلا الإنتاج

$$AB \to C$$

يمكننا أن نستبدل به

$$AB \to CD,$$
$$D \to \lambda.$$

وأما إثبات التكافؤ في هذه الحالة فهو مباشر .

٣ – ٩) العمل بالقواعد الحسَّاسة للسياق ليس دائما سهلا . وفكرة الرسول / الساعي (messenger) التي سبق تقديمها في مثال ٣ – ٢ غالبا ما تفيد في مثل هذه الحالات . وفي هذه المسألة أول خطوة نقوم بها هي إنشاء الصيغة العباراتية (sentential form) a^nBc^nD . المتغيران B, D سيعملان كعلامتين (markers) ورسولين للتأكد من أن العدد الصحيح (correct number) من كلٍّ من $a's$ و $b's$ قد تم إنشاؤه (created) في الأماكن الصحيحة . ويتم تحقيق الجزء الأول بسهولة بالإنتاجات

$$S \rightarrow aAcD \mid aBcD,$$
$$A \rightarrow aAc \mid aBc.$$

وفي الخطوة التالية يقوم المتغير B بالتحرك نحو اليمين ليقابل المتغير D ، وذلك عن طريق الإنتاجين

$$Bc \rightarrow cB,$$
$$Bb \rightarrow bB.$$

وعندما يحدث هذا ، يمكننا عندئذ إنشاء رمز d واحد ورسول عودة (a return messenger) والذي سيضع الرمز b في المكان الصحيح ويتوقف

$$BD \rightarrow Ed,$$
$$cE \rightarrow Ec,$$
$$bE \rightarrow Eb,$$
$$aE \rightarrow ab.$$

وكَحَلٍّ بديل ننشئ رمزًا d بالإضافة إلى علامةٍ D (a marker) مع رسولٍ مختلف ينشئ رمزًا b ، ولكنه يحافظ على استمرار العملية

$$BD \rightarrow FDd,$$
$$cF \rightarrow Fc,$$
$$bF \rightarrow Fb,7$$
$$aF \rightarrow abB.$$

٣ – ١٠) أسهل برهان يبدو لنا باستخدام آلة **lba** . نفرض أن لغةً ما حسَّاسة للسياق . لذلك توجد آلة **lba** – ولتكن M – تقبل هذه اللغة . إذا أُعطينا سلسلة w ، فإننا نعيد كتابتها أولا كسلسلة w^R ، ثم نطبق عليها الآلة M . ونظرا لأن $L^R = \{w : w^R \in L\}$ ، فإن

الآلة M ستقبل السلسلة w^R إذا وفقط إذا كانت $w \in L^R$. والآلة التي تعكس (reverses)سلسلة وتطبّق الآلة M هي آلة lba . ولذلك فاللغة L^R حسَّاسة للسياق .

٣ – ١١) نبدأ البرهان بآلة lba . من الواضح أنه توجد آلة lba يمكنها التعرف على أي سلسلة صيغتها wuw^R . فنبدأ تـوًّا عند النهايتين المتقابلتين (opposite ends) ، ونقارن بين الرموز لنصل إلى توافق (get a match) . ونوجد أطول سلسلة ممكنة w ، ثم نقارن طولها مع السلسلة u . ونظرا لأنه توجد آلة lba ، فلذلك تكون اللغة حسَّاسة للسياق ، وبالتالي يجب أن توجد قاعدة حساسة للسياق .

أجوبة تمرينات رقم (٤)

٤ – ١) إذا أُعطينا آلة M وسلسلة w ، فَعَدِّل الآلة M لتحصل على آلة $\hat{M}$ التي تتوقف (halts) إذا وفقط إذا كُتِبَ رمز خاص ، وليكن الرمز $\#$. ويمكننا تنفيذ ذلك بتغيير تشكيلات تَوَقُّفِ الآلة M (halting configurations of) ، بحيث أن أيًّا منها تكتب الرمز $\#$ ثم تتوقف . وهكذا فإن تَوَقُّف M يقتضي أن تكتب الآلة $\hat{M}$ الرمز $\#$ ، وكتابة $\hat{M}$ الرمز $\#$ يقتضي أن تتوقف الآلة M . وبالتالي إذا كان لدينا خوارزمية تخبرنا ما إذا كان رمزٌ مُعَيَّن a (a specified symbol) سيُكتَب إطلاقا (ever written) أم لا ، فإننا نطبقها على الآلة $\hat{M}$ حيث $a = \#$. وهذا سيَحِلُّ (solve) مسألةَ التوقُّف (halting problem) .

٤ – ٢) إذا أُعطينا (M, w) عَدِّل الآلة M إلى $\hat{M}$ ، بحيث أن (M, w) تتوقف إذا وفقط إذا كانت $\hat{M}$ تقبل لغةً بسيطةً ما ، ولتكن $\{a\}$. وهذا يمكن تنفيذه بجعل M تختبر / تتحقق (check) أولا من المدخلات ، وتتذكر ما إذا كانت المدخلات a . ثم تقوم M بتنفيذ حساباتها المعتادة . وعندما تتوقف تتحقق ما إذا كانت المدخلات a . فإن كانت a ، فإنها تقبل (accept) ، وإلا فإنها ترفض . ولذلك فإن $\hat{M}$ تقبل $\{a\}$ إذا وفقط إذا كانت M ستتوقف . والآن ننشئ آلة تيورنج بسيطة ، ولتكن M_1 ، تقبل a . فإن كانت لدينا خوارزمية تتحقق من تساوي لغتين ، فيمكننا استخدامها لنرى ما إذا كانت $L\left(\hat{M}\right) = L(M_1)$ أم لا . فإن كانت $L\left(\hat{M}\right) = L(M_1)$ فإن (M, w) ستتوقف . وإن كانت $L\left(\hat{M}\right) \neq L(M_1)$ فإن (M, w) لا تتوقف ، ويكون لدينا حل لمسألة التوقف .

٤ – ٣) إذا أُعطينا (M, w) فإننا نعدِّل M بحيث تتوقف دائما في التشكيلة (the configuration) $q_f w$. فإن كانت المسألة المعطاة قابلة للحسم ، فإننا نستطيع تطبيق الخوارزمية المفترضة على الآلة المُعَدَّلة بالتشكيلتين $q_0 w, q_f w$. وهذا سيعطينا حلا لمسألة التوقف .

٤ – ٤) نأخذ آلة تيورنج شاملة (universal) ونجعلها تحاكي (simulate) الحسابات على شريط فارغ . وكلما توقفت الحسابات المحاكاة ، فإننا نقبل (accept) آلة تيورنج التي نحاكيها . ولذلك فإن آلة تيورنج الشاملة تُعَدُّ آلة قبول (an accepter) لجميع آلات تيورنج التي تتوقف عندما تُطبَّق على شريط فارغ . ولذلك فإن المجموعة قابلة للتعديد ارتداديا .

والآن نفرض أن المجموعة ارتدادية . عندئذ ستوجد خوارزمية A تعطينا قائمة (list) بجميع آلات تيورنج التي تتوقف على شريط فارغ مُدْخَل بترتيبٍ ما لأطوال متزايدة (increasing lengths) للبرنامج . ننظر إن كانت آلة تيورنج الأصلية بين آلات تيورنج التي تولدها الخوارزمية A . ونظرا لأن طول البرنامج الأصلي ثابت / محدد (fixed) ، فإن المقارنة ستتوقف عندما نتجاوز هذا الطول . وهكذا يكون لدينا حل لمسألة التوقف على شريط فارغ .

٤ – ٥) إذا كانت العناصر المعيَّنة لمجال المسألة المحدود هي (specific instances of the problem are) $p_1, p_2, \ldots, p_n$ ، فإننا ننشئ (construct) آلة تيورنج تعمل (behaves) كما يلي :

إذا كانت المسألة $= p_1$ أَعِد القيمة : خطأ
إذا كانت المسألة $= p_2$ أَعِد القيمة : صح
$\vdots$
إذا كانت المسألة $= p_n$ أَعِد القيمة : صح

if problem = p_1 then return false,
if problem = p_2 then return true,

$\vdots$

if problem = p_n then return true.

وبغض النظر عن قيم الصحة (truth values) للعناصر المختلفة ، فإنه توجد دائما آلة تيورنج ما تعطي الإجابة الصحيحة . وتَذكَّر أنه ليس من الضروري معرفة ما هي آلة تيورنج هذه ، وإنما المهم فقط أن نضمن وجود هذه الآلة .

٤ – ٦) نفرض أنه لدينا خوارزمية تحدد ما إذا كان $L(M_1) \subseteq L(M_2)$ أم لا . يمكننا عندئذ أن ننشئ آلة M_2 بحيث أن $L(M_2) = \varnothing$ ونطبق الخوارزمية . وبعد ذلك نرى أن $L(M_1) \subseteq L(M_2)$ إذا وفقط إذا كانت $L(M_1) = \varnothing$. ولكن هذا يناقض نظرية ٤ – ٣ ، نظرا لأنه يمكننا إنشاء الآلة M_1 من أي قاعدة معطاة G_1 .

٤ – ٧) إذا أخذنا $L(G_2) = \Sigma^*$ ، فإن المسألة تصبح هي سؤال نظرية ٤ – ٣ ، وبالتالي فهي غير قابلة للحسم .

٤ – ٨ – أ) باستخدام نظرية رايس : نظرا لأنه توجد بعض القواعد التي تحقق الشرط $L(G) = L(G)^*$ ، وقواعد أخرى لا تحققه ، فإن عدم القابلية للحسم تنتج من نظرية رايس .

ب) باستخدام المبادئ الأولية : الإثبات هنا أصعب قليلا . خذ مسألة التوقف (M, w) وعدِّلها (بناءً على خطوات نظرية ٤ – ٤) ، بحيث أنه إذا توقفت (M, w) فإن $\hat{M}$ ستقبل $\{a\}^*$ ، وإذا لم تتوقف (M, w) فإن $\hat{M}$ تقبل $\varnothing$. ومن الآلة $\hat{M}$ نحصل على القاعدة $\hat{G}$ بعملية الإنشاء (construction) التي تؤدي إلى نظرية ٣ – ٧ . فإذا كانت $L\left(\hat{M}\right) = \{a\}^*$ فإن $L\left(\hat{G}\right) = L\left(\hat{G}\right)^* = \{a\}^*$. ولكن إذا كانت $L\left(\hat{M}\right) = \varnothing$ فإن $L\left(\hat{G}\right) = \varnothing$ وكذلك $L\left(\hat{G}\right)^* = \{\lambda\}$. ولذلك فإذا كانت هذه المسألة قابلة للحسم ، فمعنى ذلك أننا سنستطيع الحصول على حل لمسألة التوقف .

٤ – ٩) أحد حلول PC - هو $w_3w_4w_1 = v_3v_4v_1$. لا يوجد حل MPC - نظرا لأن إحدى السلسلتين سيكون لها سابقة 001 (a prefix) ، والأخرى 01 .

٤ – ١٠) إذا كان $|w_i| > |v_i|$ أو $|w_i| < |v_i|$ لجميع قيم i ، فمن الواضح أنه لن يكون هناك حل . وإذا لم يتحقق هذا الشرط ، فإما أن يكون $|w_i| = |v_i|$ لقيمةٍ ما i ، وهذه لها حل بسيط / تافه (a trivial solution) ، أو أنه يجب أن توجد قيمة i

وقيمة k بحيث أن $|w_j| > |v_j|$ و $|w_k| < |v_k|$. وفي الحالة الأخيرة فإن المسألة PCP سيكون لها حل $w_j^r\, w_k^s$ ، حيث $r = |v_k| - |w_k|, s = |w_j| - |v_j|$.

٤ – ١١) المسألة غير قابلة للحسم . ولو كانت قابلة للحسم لكان لدينا خوارزمية لحسم المسألة MPC - الأصلية . فإذا أُعطينا $w_1, w_2, \ldots, w_n$ ، فإننا نقوم بتكوين $w_1^R, w_2^R \ldots, w_n^R$ واستخدام الخوارزمية المفترضة . ونظراً لأن $w_1 w_i \ldots w_k = \left(w_k^R \ldots w_i^R w_1^R\right)^R$ ، فإن المسألة - MPC الأصلية يكون لها حل إذا وفقط إذا كانت المسألة - MPC الجديدة لها حل .

٤ – ١٢ – أ) أوجد منتصف السلسلة (middle of the string) ، ثم تحرك خلفا وأماما لمواءمة (matching) الرموز . والجزءان يستغرقان $O(n^2)$ من التحركات .

ب) احسب عدد الرموز . فإذا تم الحساب أحاديا (count is done in unary) ، فإن هذه عملية (operation) $O(n)$. وبعد ذلك اكتب النصف الأول على الشريط الثاني . وهذه أيضا عملية $O(n)$. وأخيرا يمكننا المقارنة بتحركات عددها $O(n)$.

ج) يمكننا تخمين منتصف السلسلة (middle of the string) بطريقة غير محددة (nondeterministically) بحركة واحدة . ولكن المواءمة (matching) ستظل تستغرق $O(n^2)$ من الحركات .

د) قم بتخمين منتصف السلسلة ، ثم تحرك كما في الجزء ب) . والجهد الكلي المطلوب هو $O(n)$.

أجوبة تمرينات رقم (٥)

٥ – ١) باستخدام الدالة *subtr* المُعَرَّفة في مثال ٥ – ٣ نحصل على الحل

$$greater\ (x, y) = subtr\ (1, subtr\ (1, subtr\ (x, y)))$$

٥ – ٢)

$$g\ (x, y) = mult\ (x, g\ (x, y - 1))$$
$$g\ (x, 0) = 1.$$

٥ – ٣ – أ)

$$\begin{aligned} A(1, y) &= A\ (0, A\ (1, y - 1)) \\ &= A\ (1, y - 1) + 1 \\ &= A\ (1, y - 2) + 2 \\ &\vdots \\ &= A\ (1, 0) + y \\ &= y + 2. \end{aligned}$$

ب) بعد حصولنا على نتائج الجزء أ) من السؤال يمكننا استخدام الاستقراء (induction) لإثبات المتطابقة التالية . نفرض أنه لقيم $y = 1, 2, \ldots, n - 1$ لدينا $A(2, y) = 2y + 3$. بناءً عليه نحصل على

$$\begin{aligned} A(2, n) &= A\ (1, A\ (2, n - 1)) \\ &= A\ (1, 2n + 1) \\ &= 2n + 3 \quad [\text{ من الجزء أ) }] \end{aligned}$$

ونظرًا لأن

$$\begin{aligned} A\ (2, 0) &= A\ (1, 1) \\ &= 3 \end{aligned}$$

يكون لدينا أساس ، وتكون المعادلة صحيحة لجميع قيم y .

٥ – ٤) إذا كانت $2^x + y - 3 = 0$، فإن $y = 3 - 2^x$. وقيم x الوحيدة التي تعطي قيمة موجبة لـ y هي الصفر 0 والواحد 1 . وبالتالي فإن مجال μ هو $\{0, 1\}$ ، معطيا قيمةً صغرى لـ y تساوي $y = 1$. ولذلك فإن

$$\mu y\,(2^x + y - 3) = 1$$

٥ – ٥) استخدم $C_T = \{a, b, c\}, C_N = \{x\}, A = \{x\}$. و x غير الطرفية (nonterminal x) تُستخدم كحد (a boundary) بين الجانب الأيسر والجانب الأيمن للسلسلة الهدف (target string)، ويتم بناء السلسلتين $w's$ في الوقت نفسه (simultaneously) بالإنتاجات

$$V_1xV_2 \rightarrow V_1axV_2a \,|V_1bxV_2b|\, V_1cxV_2c$$

وأخيرا نزيل x (remove) بالإنتاج

$$V_1xV_2 \rightarrow V_1V_2$$

٥ – ٦) في كل خطوة التعريف / التقديم / التعرُّف (identification) الوحيد الممكن للمتغير V_1 يكون مع / ضمن السلسلة المشتقة بأكملها (the entire derived string) . وهذا يؤدي إلى مضاعفة السلسلة (doubling the string)، وبالتالي تكون اللغة المطلوبة هي

$$L = \left\{a^{2^n} : n \geq 1\right\}$$

٥ – ٧) أحد الحلول هو :

$$V_1 * V_2 = V_3 \rightarrow V_1 1 * V_2 = V_3V_2,$$
$$V_1 * V_2 = V_3 \rightarrow V_1 * V_2 1 = V_3V_1.$$

فمثلا

$$1 * 1 = 1 \Rightarrow 11 * 1 = 11 \Rightarrow 11 * 11 = 1111,$$

وهكذا .

٥ – ٨)

$$P_1 : S \to S_1S_2,$$
$$P_2 : S_1 \to aS_1,\ S_2 \to aS_2,$$
$$P_3 : S_1 \to bS_1,\ S_2 \to bS_2,$$
$$P_4 : S_1 \to \lambda,\ S_2 \to \lambda.$$

٥ – ٩) الحل هنا يذكِّرنا باستخدام الرُسل (messengers) مع القواعد الحسَّاسة للسياق (context-sensitive grammars).

$$ab \to x,$$
$$xb \to bx,$$
$$xc \to \lambda.$$

٥ – ١٠) أساسا / مبدئيا (in principle) يجب إعادة كتابة (rewriting) كل رمز (symbol) في كل خطوة . ويمكننا أن نتحايل على هذا بالإنتاج $a \to a$. ولذلك فيمكننا إعادة الكتابة بحيث أنه في كل خطوة تُضاف a واحدة ، وهذا يعطينا اللغة $L(aa^*)$.

أجوبة تمرينات رقم (٦)

٦ – ١) اختيار الخوارزمية في موضوع الترتيب / الفَرْز أمر هام . ودرجة التعقيد الزمنية للطرق البسيطة كطريقة الترتيب الفقاعي (bubblesort) هي ($O(n^2)$ – time complexity) . أما درجة التعقيد الزمنية لأكفأ خوارزميات الترتيب / الفَرْز فهي ($O(n \log n)$).

٦ – ٢) $(x_1 \vee x_3) \wedge (x_2 \vee x_3)$

٦ – ٣) إذا زادت / نَمَتْ تشكيلة واحدة (one configuration grows) ، فإن المعلومات الموجودة على الشريط يجب تحريكها (to be moved) . نفرض أنه يجب عمل إزاحة يُمنى (a right shift) . اتجه إلى النهاية اليمنى للشريط ، وحَرِّك كل رمز (symbol) في المنطقة النشطة (الفعالة) (active area) خليةً واحدةً إلى اليمين . وهذا يتطلب عدد $O(n\,k^n)$ من الحركات . فإذا كانت كل تشكيلة (configuration) من التشكيلات التي عددها k^n تزيد / تنمو في حركة مفردة (in a single move) ، فإن العملية كاملةً تتطلب $O(n^3\,k^{2n})$ من الحركات (moves) . ونظرا لأن هذه القيمة محكومة بالقيمة $O(k^{3n})$ (dominated by) ، فإن النتيجة التي وصلت إليها النظرية لا تتأثر بإهمال النقطة المشار إليها .

٦ – ٤) هذه نتيجة مباشرة (an immediate consequence) لنظرية ٦ – ٤ .

٦ – ٥) يميل المرء إلى أن يقول شيئا مثل : " بطريقة غير محددة (nondeterministically) اختر V_i كالرأس الأولى ، و V_j كالرأس الثانية ، وهكذا " ، ولكن هذا غير صحيح . فبينما عدم التحديد (nondeterminism) يقتضي اختيارا (implies a choice) ، لكن الاختيار في كل حركة هو من عدد محدود من الخيارات المتاحة (a limited number of options) . ونظراً لأن قيمة المؤشر i في V_i هي قيمة كبيرة كبرا اختياريا (is arbitrarily large) ، لا يمكننا إجراء ذلك . وهناك طريقة أفضل من ذلك ، وتتلخص في الخطوات التالية :

(١) أُنْشِئ قائمة على الأعداد $1, 2, \ldots, n$ بالاصطلاح الأحادي (in unary notation) . وطول القائمة هو $O(n^2)$ ، وبالتالي فيمكن تنفيذ ذلك في فترة زمنية $O(n^2)$ time .

(٢) قم بمسح القائمة (scan the list) ، وبطريقة غير محددة (nondeterministically) اختر (select) عددا واحدا . وإذا تم اختيار عدد ، فَأَضِفْه إلى التبديل (the permutation) ، وأَزِلْهُ (remove it) من القائمة (list) .

(٣) كَرِّر الخطوة (٢) عدد n من المرات .

وهكذا فيمكننا تنفيذ العملية كلها (the whole process) في وقت time $O(n^3)$.

٦-٦) خذ المخطط البياني للمسألة **HAMPATH** ، وأكمله (complete it) . قم بإعطاء أوزان لأحرف المخطط : **0** للأحرف الموجودة (existing edges) ، و **1** للأحرف الجديدة (new edges) . طَبِّق الخوارزمية **TSP** حيث $k = 0$. فإن كان هناك حل (a solution)، فعندئذ يوجد مسار " هاملتوني " .

أسماء بعض المراجع في نظرية الحوسبة والأوتوماتية واللغات الشكلية

[1] An Introduction to the Theory of Formal Languages and Automata by Willem J. M. Levelt (2008)

[2] An Introduction To Automata Theory & Formal Languages by Adesh K. Pandey (2007)

[3] An Introduction To Formal Languages and Automata Theory by Peter Linz (2011)

[4] Theory of Finite Automata With an Introduction to Formal Languages by John Carroll and Darrell Long (1989)

[5] Automata and Formal Languages: An Introduction by Dean Kelley (1995)

[6] A Second Course in Formal Languages and Automata Theory by Jeffrey Shallit (2008)

[7] Introduction to Automata Theory, Languages, and Computation (3rd Edition) by John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman (2013)

[8] Automata and Languages: Theory and Applications by Alexander Meduna (2000)

[9] Introduction to Languages, Machines, and Logic by Alan P. Parkes (2002)

[10] A Concise Introduction to Languages and Machines by Alan P. Parkes (2008)

[11] Algebraic Theory of Automata & Languages by Masami Ito (2004)

[12] Automata Theory with Modern Applications by James A. Anderson (2006)

[13] Automata Theory and its Applications (Progress in Computer Science and Applied Logic (PCS)) by Bakhadyr Khoussainov and Anil Nerode (2001)

[14] Gentle Introduction to the Theory of Computing (Concepts, Techniques and Understandings of Computing, Automata and Formal Languages) by Zamir Bavel (1996)

[15] Problem Solving in Automata, Languages, and Complexity by Ding-Zhu Du and Ker-I Ko (2001)

[16] An Introduction to Formal Languages and Machine Computation by Song Y. Yan (1996)

[17] Introduction to the Theory of Computation by Michael Sipser (2012)

[18] Formal Languages and Automata Theory by Nagpal (2012)

[19] Switching and Finite Automata Theory by Zvi Kohavi and Niraj K. Jha (2009)

[20] Introduction To Automata Theory, Formal Languages and Computation by Shyamalendu Kandar (2013)

[21] Introduction to Theory of Automata, Formal Languages and Computation by Debidas Ghosh (2013)

[22] Modern Applications of Automata Theory by Deepak D'souza and Priti Shankar (2012)

[23] Theory of Automata And Computation by Adesh K. Pandey (2015)

[24] Formal Languages, Automata and Numeration Systems by Michel Rigo (2014)

[25] Theory of Computation by George Tourlakis (2012)

دليل المصطلحات العربية والإنجليزية
Index

فيما يلي قائمة بأسماء المصطلحات العلمية باللغة الإنجليزية مرتبة ترتيبا أبجديا ، والمصطلحات المقابلة لها باللغة العربية ، مع أرقام الصفحات التي ظهرت فيها هذه المصطلحات .

argument	٤، ٣٠	وسيط / حُجَّـة
array	٢	منظومة
assignment of values	١٥٨	إسناد القيم
asymptotic growth rate	١٥٥	مُعَدَّل التزايد / النمو التقاربي
automaton	٢	آلة ذاتية الحركة
axiom	١٤١	موضوعة / مُسَلَّمَة / فَرَضِيَّة
backtracking algorithm	٥٤	خوارزمية لتعقُّب المسار العكسي (من الخلف للأمام)
basic Turing machine	٣٦	آلة تيورنج الأساسية
behavior (of an algorithm)	١٥٤	سلوك (خوارزمية)
blank	١١	فراغ
blank – tape halting problem	١٠٨	مسألة التوقف على شريط فارغ
block diagram	٢٢	مخطط (بياني / قالبي)
boolean operator	١٥٧	مؤثِّر بُـولي
branching factor	٥٦	معامل التفرُّع
bruteforce method	١٦٧	طريقة قسرية
bubblesort algorithm	٢٠٢	خوارزمية الترتيب الفقاعي
building blocks	٢٢	قوالب البناء
calling program	٢٦	البرنامج المستدعِي
capacity	٥	سعة
category	٣٥	طبقة / دائرة
cell	٢	خلية
characterization	١٥٤	توصيف / تحديد
Church's thesis	١٣١	رسالة " تشيرش "
clause	١٥٨	عبارة
Clique problem	١٧٢	مسألة العُصْبَة
CNF (conjunctive normal form)	١٥٧	الصيغة القياسية / المعيارية العَطْفِيَّة
coalescing	١٨	دَمْج
combining Turing machines	٢٢	الجَمْع بين آلات تيورنج
comparer	٢٣	جهاز المقارنة

comparing machine	٢٣	آلة مقارنة
complement	٣٤	مكمِّـل
complexity theory	١٥٦	نظرية التعقُّد / التعقيد
compose	٢٩	يُركِّب / يكوِّن
composition	١٣٣	التركيب
computability	٩٩	القابلية للحوسبة / للحساب
computable	١٧	قابلة للحساب
computation	١١	حَوْسبة / حِسْبة / عملية حسابية
computational complexity	١٥٣	درجة التعقيد الحاسوبية
configuration	٥، ٣٧	هيئة / تشكيلة / حالة تكوينية
configuration tree	٥٦	شجرة التشكيلات
conjecture	٢٩	تخمين
conjunctive normal form (CNF)	١٥٧	الصيغة القياسية / المعيارية العَطْفِيَّة
context – sensitive grammar	٨٨	قاعدة حسَّاسة للسياق
context – sensitive language	٨٩	لغة حسَّاسة للسياق
context – free language	١	لغة حرة السياق
control unit	٣	وحدة تحكم
convention	٨	اصطلاح
Cook's theorem	١٧٧	نظرية " كوك "
copier	٣٣	جهاز نسخ
countable set	٥٩	مجموعة قابلة للعد
counterexample	٢٩	مثال مناقض
criterion	٦٢	معيار
current state	٤	الحالة الحالية
decidability	٩٩	القابلية للحسم / للتقرير / للتحديد
delimit the field	٥٢	يحدِّد (يبيِّن حدود) المجال
derivation string	١١٩	سلسلة الاشتقاق
detect	١٥	يكتشف
deterministic machine	٨	آلة محدَّدة
diagonalization	٧٥	إقطار

digital computer	١٧	حاسوب رقمي
divisor	٦٥	القاسم
domain	١٧	مجال
edge	٦	حرف
efficiency of computation	١٥٤	كفاءة الحوسبة
emptiness algorithm	١١٢	خوارزمية الخواء
empty string	١٣	السلسلة الخالية
encapsule	٢٢	يغلِّف
endless loop	١١	عُرْوَة لا نهائية
enumeration procedure	٦١	طريقة / إجراء تعديد / عَدّ
equivalence of classes of automata	٣٦	تكافؤ طبقات الآلات
equivalent automata	٣٦	آلات متكافئة
erase a symbol	١٥	يمسح رمزا
erasing program	٢٣	برنامج محو
exhaustive approach	١٥٨	الطريقة الشاملة / الاستنفادية
exhaustive search	١٢٧	البحث الشامل
exponential complexity	١٥٨	درجة تعقيد أُسِّيَّة
extended hierarchy	٩٥	تدرج هرمي موسّع
extended L - system	١٤٩	نظام - L المُوَسَّع / الممتد
extension	١٦	امتداد
extensive set	٢٣	مجموعة مُوَسَّعة
extraneous blanks	٨٥	فراغات متفرقة
final state	٤	حالة نهائية
finite accepter	٩٦	آلة قبول محدودة
finite automaton	١	آلة محدودة
finite memory	٥	ذاكرة محدودة
finite set	٤	مجموعة محدودة
finiteness algorithm	١١٤	خوارزمية المحدودية
general – purpose computer	٥٩	حاسوب متعدد الأغراض
generality	٩٢	العمومية

graph	٦	مخطط بياني
growth pattern	١٤٨	نمط / نموذج نمو
growth rate	١٣٨	مُعَدَّل النمو
halt	٥	يتوقف / توقُّف
halt state	٥	حالة تَوَقُّف
halt state	١١	حالة تَوَقُّف
halting configuration	١١٤	تشكيلة تَوَقُّف
halting problem	٩٩	مسألة توقف
halting state	١١٣	حالة تَوَقُّف
Hamiltonian path	١٧٠	مسار " هاملتوني "
hardware	١٥٣	المعدات / المكوِّنات المادية
hierarchy	٧١	تسلسل هرمي
high – level description	٢٢	وَصْف عالي المستوى
high level language	١٣	لغة عالية المستوى
hypothesis	٢٩	فَرْض
identity	١٥١	متطابقة
implementation	١٧	تنفيذ
implication	٦٥	اقتضاء
induction	٨٧	الاستقراء
infinite tape	٣٦	شريط لا نهائي
inherent	١٣١	ذاتي / كامن
initial configuration	٦	هيئة تكوينية ابتدائية
initial state	٤	حالة ابتدائية
input alphabet	٤	مجموعة أبجدية للإدخال
input file	٢	ملف إدخال
instantaneous description	٣٧	وصف لَحْظِي
instantaneous description	٨	وَصْف لَحْظِي
instruction stream	٢٢	سيل التعليمات
interact	٢٦	يتفاعل
internal state	٤	حالة داخلية

interpretation	٤	تفسير
intractable	١٦٨	صعبة المعالجة / عَسِرة المعالجة
invertible	٩٣	قابل للعكس
keep track of	١٢	يتعقب
key number	١٥٦	عدد مفتاح
L – system	١٤٨	نظام - L
label	٦	يُسمِّي / اسم / عنوان
language accepter	١١	آلة قبول للغة
leading blanks	٨٣	فراغات أمامية
linear – time algorithm	١٥٨	خوارزمية خَطِّيَّة الزمن
linear bounded automata (lba)	٦٤	آلات مقيَّدة / محدودة خطية
linear search	١٥٦	بحث خَطِّي
literal	١٥٨	حَرْفِيّ
logic operator	١٥٧	مؤثِّر منطقي
machine language	٢٢	لغة الآلة
macroinstruction	٢٤	تعليمة ماكرو / إجمالية
Markov algorithm	١٤٦	خوارزمية " ماركوف "
match	١٦، ٧٤	توافق / توائم / يوائم
matrix grammar	١٤٥	قاعدة مصفوفية
mechanism	٣	آلية
membership algorithm	٧٢	خوارزمية عضوية / انتماء
messenger	٩٠	رسول
mimic	٣٧	يحاكي / يقلِّد
minimalization operator	١٣٩	مؤثِّر الأصغرِيَّة
Modified Post Correspondence Problem (MPCP)	١١٧	مسألة " بوست " المُعَدَّلة للمقابلة
monus ($\dot{-}$)	١٣٤	مؤثر / عامل لنوع خاص من الطرح
move	٤	حركة
MPCP (Modified Post Correspondence Problem)	١١٧	مسألة " بوست " المُعَدَّلة للمقابلة
multidimensional machine	٥٠	آلة متعددة الأبعاد

multihead Turing machine	٦٨	آلة تيورنج متعددة الرؤوس
multiple tracks	٤١	مسارات متعددة
multiplicand	٢٧	العدد المضروب فيه
multiplication machine	٢٧	آلة ضرب
multiplier	٣٣	جهاز ضرب
multitape machine	٤٦	آلة متعددة الأشرطة
multitrack machine	٦٤	آلة متعددة المسارات
negation operator	١٥٧	مؤثِّر النفي
nested structure	٩٦	بنية متداخلة
nonblank symbol	١٧	رمز غير فراغي
noncontracting	٨٨	غـــير متقلـــص / غـــير منكـــمش (نوع من القواعد)
nondeterminism	٣٥	عدم التحديد
nondeterministic Turing machine	٥٣	آلة تيورنج غير محدَّدة
nonerasing Turing machine	٦٧	آلة تيورنج غير الماحية
nonfinal state	١٢	حالة غير نهائية
nonterminal constants	١٤١	ثوابت غير طرفية
normal form	٨٨	صيغة قياسية
NP – complete language	١٧٧	لغة تامة / كاملة - NP
NP - completeness	١٧٧	التمام / الكمال - NP
npda (nondeterministic pushdown accepter)	٦٩	آلة قبول غير محددة ذات دفع لأسفل
off – line machine	٤٤	آلة مفصولة عن الخط
open problem	١٧٩	مسألة مفتوحة للبحث
order – of – magnitude analysis	١٥٥	تحليل حدود القيم
orders – of – magnitude expressions	١٥٣	تعابير حدود القيم
outcome	٢٢	نتيجة
partial function	٤	دالة جزئية
pass	١٤	مرور / دورة
PCP (Post Correspondence Problem)	١١٥	مسألة " بوست " للمقابلة

performance (of an algorithm)	١٥٤	أداء (خوارزمية)
permutation	١٨٠	تبديل
phrase	٢٤	عبارة
phrase – structure grammar	١٤٥	قاعدة ذات بنية عباراتية / جُمْلية
polynomial – time algorithm	١٧٧	خوارزمية حدودية الزمن
polynomial – time reduction	١٧٣	الاختزال في فترة زمنية حدودية
Post Correspondence Problem (PCP)	١١٥	مسألة " بوست " للمقابلة
Post system	١٤٠	نظام " بوست "
powerful	٢٨	ذو قدرة عالية
predecessor function	١٣٤	الدالة السابقة / المقدِّمة / القَبْلِيَّة
primitive	٢٨	بدائي
primitive recursion	١٣٣	الارتداد البدائي
primitive recursive function	١٣٥	دالة ارتدادية بدائية
problem	١٠٠	مسألة
procedure	٢٩	إجراء
processing unit	٥	وحدة تشغيل
projector function	١٣٢	دالة الإسقاط
proper order	٦٢	ترتيب صحيح / سليم / مضبوط
proper subset	١	مجموعة جزئية فعلية / صحيحة
pseudo code / pseudocode	٢٢	(كـود / برنـامج) (شـبيه / زائـف) / شبه كود
pushdown automata	١	آلات ذوات الدفع لأسفل
pushdown automata (pda)	١٦	آلات الدفع لأسفل
queue	١	طابور
queue automaton	٦٨	آلة طابور ذاتية الحركة
quotient	٦٠	كسر اعتيادي
range	١٣١	مدى
read – write head	٣	رأس القراءة والكتابة
real – life performance	١٥٥	الأداء الفعلي في الواقع العملي
recursive function	١٣١	دالة ارتدادية

recursive language	٧٢	لغة ارتدادية
recursive program	١٣٧	برنامج ارتدادي
recursively enumerable language	٧٢	لغة قابلة للتعديد ارتداديا
reduce	١٠٧	يختزل
reduction technique	١٠٧	طريقة الاختزال
reference point	٤٢	نقطة إسناد / نقطة مرجع
refine (a description)	٢٢	يَصْقل / يُفصِّل / يُنقِّح (وَصْفا)
region separator	٢٦	فاصل مناطق
regular language	١	لغة منتظمة
replicating	٥٤	تكرير / نسخ
reprogrammable machine	٣٦	آلة قابلة لإعادة البرمجة
resource requirements	١٥٣	متطلبات الموارد
rewriting system	١٤٤	نظام إعادة الكتابة
Rice's theorem	١١٥	نظرية " رايس "
satisfiability problem (SAT)	١٥٦	مسألة التَّحَقُّقِيَّة
satisfiable	١٧٨	تَحَقُّقِي
satisfiable expression	١٥٨	تعبير تَحَقُّقِي
scheme	٢	مخطط
scratch space	٦٦	حيِّز للعمل والتسويد
secondary storage	٥	تخزين ثانوي
semi – infinite tape	٤١	شروط نصف لا نهائي
sensing	٥	الإحساس
sentential form	٨٥	الصيغة الحارسة
sentential form	٨٨	صيغة عباراتية
sequence of configurations	٨	متتابعة تشكيلات
sequence of moves	٦	متتابعة تحركات
signal	٢٢	إشارة
simulating machine	٣٩	آلة محاكية
simulation	٣٧	محاكاة
single out	١٤٦	يُفْرد

situation	٤	وَضْـع
size of a problem	١٥٤	حجم / سعة المسألة
sketch	٢٩	مخطط
software	١٥٣	البرمجيات
sorting problem	١٥٤	مسألة الترتيب / الفَرْز
space complexity	١٥٣	درجة التعقيد المكانية
stack	٣٤	رصَّـة
stage	٦	مرحلة
standard machine	٨ ،٢	آلة قياسية
state	٤	حالة
state – entry problem	١٠٧	مسألة دخول الحالة
state of control	٥	حالة التحكم
stay – option	٣٨	اختيار البقاء (في الموضع)
storage	٢	مخزن / تخزين
storage device	٣٥	وسيلة تخزين
string	١١	سلسلة
string manipulations	٢٩	معالجات للسلاسل
structure	١٦	بنية
subprogram	٢٥	برنامج فرعي
subscript	١٠	مؤشِّر / رمز سُفْلِي
subsequent	٦	تالي
subtracter	٣٣	جهاز طرح
successor function	١٣٢	الدالة التابعة / التالية / البَعْدِيَّة / اللاحقة
suspend	٢٦	يوقف
symbol	٤	رمز
synthesis	١٩	تركيب
tape	١١	شريط
tape alphabet	٤	مجموعة أبجدية للشريط
task	٢٢	مهمة

technical	١٣	فَنّي / تقني
temporary storage	١	تخزين مؤقت
term	١٥٧	حَدّ
terminal constants	١٤١	ثوابت طرفية
terminal productions	١٤٦	إنتاجات طرفية
terminal string	١٤٦	سلسلة طرفية
terminals	٨١	طَرَفيات / رموز طَرَفية
terminate	٥	يتوقف / ينتهي / ينهي
time complexity	١٥٣	درجة التعقيد الزمنية
total function	١٣٥	دالة كُلِّيَّة
trace	٢١	يتتبع
track	٤١	مسار
tractability	١٧٧	سهولة المعالجة / يُسر المعالجة
tractable	١٦٨	طَيِّعَة / سهلة المعالجة
trailing blanks	٨٣	فراغات خلفية
transducer	١٧	محوِّل للطاقة
transition	١٤	انتقال
transition function	٤	دالة انتقال
transition graph	٦	مخطط بياني للانتقالات
transition graph	٦	مخطط بياني للانتقالات
traveling salesman problem (TSP)	١٨١	مسألة البائع المسافر
triplet	٤٠	ثلاثية
TSP (traveling salesman problem)	١٨١	مسألة البائع المسافر
Turing hypothesis	٥٩	فرضية تيورنج
Turing machine	١	آلة تيورنج
Turing's thesis	٢٨	رسالة تيورنج
two – dimensional tape	٥١	شريط ثنائي البعد
typical case	٧	حالة نمطية
unary notation	١٨	الاصطلاح الأحادي
unary representation	١٤٣	تمثيل أحادي

كتب للمؤلف

في الرياضيات وعلم الحاسوب

١ – برمجة الحاسب بلغة الفورتران ، ط ٤ ، دار القلم .. الكويت ١٩٩٢ .

٢ – مقدمة في نظرية المعلومات ، ط ٢ ، دار القلم .. الكويت ١٩٩٣ .

٣ – الشبكات الرقمية ، دار القلم .. الكويت ١٩٨٦ .

٤ – التحليل العددي ، دار القلم .. الكويت ١٩٨٨ .

٥ – الجبر الخطي ، ط ٢، دار القلم .. الكويت ١٩٩٥ .

٦ – برمجة الحاسب بلغة الباسكال ، ط٢ ، دار القلم .. الكويت ١٩٩٩ .

٧ – البرمجة المتقدمة بلغة الباسكال ، مع د. حمزة رشوان، دار القلم .. الكويت ١٩٩٤.

٨ – الدوائر المتكاملة الرقمية (ترجمة) ، دار القلم .. الكويت ١٩٩٣.

٩ – الخوارزميات والبرمجة بلغة الباسكال ، دار القلم .. الكويت ١٩٩٧.

١٠– بنى المعطيات ، مع د. حمزة رشوان ، دار القلم .. الكويت ١٩٩٨.

١١– الحلول العددية للمعادلات التفاضلية العادية ، دار القلم .. الكويت ٢٠٠٠ .

١٢– الحلول العددية للمعادلات التفاضلية الجزئية ، دار القلم .. الكويت ٢٠٠١ .

١٣– برمجة الحاسب بلغة C++ ، دار القلم .. الكويت ٢٠٠٢ .

١٤– البرمجة المتقدمة بلغة C++ ، دار القلم .. الكويت ٢٠٠٣ .

١٥– الرياضيات المتقطعة في علم الحاسوب ، مكتبة الفلاح .. الكويت ٢٠٠٥ .

١٦– هياكل البيانات بلغة C++ ، مع د. حمزة رشوان ، مكتبة الفلاح .. الكويت ٢٠٠٦ .

١٧– البرمجة المتقدمة بلغة C ، دار اقرأ .. الكويت ٢٠٠٦ .

١٨ – برمجة الحاسوب بلغة C ، دار اقرأ .. الكويت ٢٠٠٧ .

١٩– نظم الحاسوب ، مكتبة الفلاح .. الكويت ٢٠١١ .

٢٠ – نظرية الحوسبة والأوتوماتية واللغات الشكلية ، مكتبة الفلاح .. الكويت ٢٠١١ .

٢١ – الطرق العددية والتحليل العددي ، مكتبة الفلاح .. الكويت ٢٠١٢ .

٢٢ – النظرية المتقدمة في الحوسبة والأوتوماتية واللغات الشكلية ، ذات السلاسل .. الكويت ٢٠١٥.

٢٣ – تصميم وتحليل خوارزميات الحاسوب، ذات السلاسل .. الكويت ٢٠١٦.

محتويات الكتاب

Advanced Theory of Computation, Automata, and Formal Languages

Dr. Abu-Bakr Ahmad El-Sayed

Dept. of Computer Science
University of Kuwait