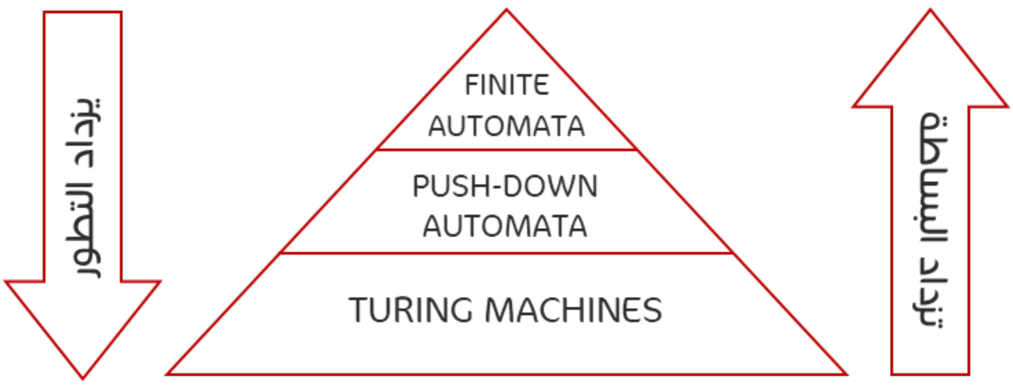
نظرية الأتومات
والأتومات المنتهي
Finite Automata (FA)

د. محمد الأحمد

نظرية الأتومات (Automata Theory)

هي فرع من فروع علوم الحاسب, يدرس الأجهزة المجردة (abstract devices) وهي أجهزة غير فيزيائية, تقدم هذه الأجهزة خدمات حاسوبية عديدة (جمع, طرح ,حسابات, عمليات منطقية, ...), تساعد هذه الأجهزة في حل مشاكل تواجهنا حيث تقدم نمذجة للواقع أي تقوم بتقديم model والذي هو تبسيط للواقع بهدف فهم المشكلة (مثال: المخططات, الخرائط, الرسومات التوضيحية هي جميعها model لأنها تساعدنا على تبسيط وفهم الواقع).

المادة بشكل عام ستتناول مختلف أنواع هذه الأجهزة (سنقوم بتسمية الأجهزة بالأتومات) وما الحلول التي تقدمها, وسنبدأ بالأجهزة من المستوى الأبسط إلى الأكثر تعقيداً حسب هرمية Chomsky للأتومات والتي هي كالتالي:



ما هي أهمية نظرية الأتومات وتطبيقاتها؟

كل نموذج من الأتومات يلعب دوراً هاماً في العديد من النواحي التطبيقية,فالأتومات المنتهي يُستخدم في المعالجة النصيّة text processing وفي المترجمات compilers وفي تصميم العتاديات hardware designing وبعض استخدامات push-down automata تكمن في اللغات البرمجية والذكاء الصنعي...

اللغات الصورية (Formal Language):

تشمل جميع اللغات (الطبيعية/البرمجية/البشرية/...).

تتألف اللغة الصورية من مجموعة من الكلمات (word/string) بحيث تكون حروف هذه الكلمات (symbols) مأخوذة من أبجدية معينة (alphabet), وتخضع هذه الكلمات لقواعد معينة لتكوّن نهايةً اللغة الصوريّة.

كل نوع من الأتومات يعرف نوع من اللغات التي يقبلها (كما سنوضح لاحقاً).

بعض أنواع الأتومات

Finite Automata

- هي أبسط أنواع الأجهزة.
- تحتوي على مجموعة منتهية (finite) من الحالات والتي يتم الانتقال فيما بينها عن طريق حدث أو دخل معين.
- تعرّف هذه الأجهزة اللغة المنتظمة (Regular Language) وهي أبسط أنواع اللغات.
- تقوم بحل مشاكل بسيطة غير معقدة.
- ذاكرتها محدودة أو لا تحتاج إلى ذاكرة أساساً,فهي لا تخزن أي حالة سابقة للأتومات وتعرف الحالة الراهنة فقط.
- تستخدم لنمذجة الأجهزة الحاسوبية البسيطة (small computers).

Push-down Automata

- تقوم بتعريف لغات خارج السياق (context-free language) مثل اللغات البرمجية.
- ذاكرتها غير محدودة ولكن الوصول إلى هذه الذاكرة يكون وصولاً مقيداً بشروط معينة (مثال: قد يتم التعامل مع مكدس stack فيكون الوصول إلى الذاكرة مقيّداً من جهة واحدة (والتي هي قمة المكدس)).
- تستخدم لنمذجة ال parsers (والذي هو برمجية معينة من مكونات الcompiler يقوم ببناء بنية معطيات معينة(غالباً يقوم ببناء parsing tree أو بنية هرمية أخرى) ويقوم بإعطاء تمثيل بنيوي للبرنامج المُدخَل من خلال البنية التي استعملها والذي يستخدم للتحقق من الكتابة الصحيحة للبرنامج syntax check).

Turing Machines

- أرقى أنواع الأجهزة من حيث التطور حيث أنها قادرة على حل أي مشكلة حاسوبية (أي مشكلة هي قابلة للحوسبة إذا استطعنا أن نصمم Turing Machine لها).
- تقوم بتعريف اللغات السياقية(context languages) مثل اللغات الطبيعية.
- تستخدم لنمذجة أي جهاز حاسوبي.

الفرق بين اللغات السياقية (context languages) واللغات خارج السياق (context-free language):

- اللغات الطبيعية (مثل اللغة العربية /اللغة الإنكليزية/ .....) هي جميعها لغات سياقية حيث أنه من الممكن ان تأخذ كلمة من اللغة عدة معاني ودلالات حسب موقعها من الجملة وحسب سياقها، **مثال : عبارة "مُتُّ من الضحك"**،لا نعني بكلمة "مُتُّ" بحالة الموت الحرفي وإنما سياق هذه الجملة يدل على حالة الضحك الشديد.
- أما اللغات خارج السياق لا يمكن لكلمة من كلماتها أن تأخذ دلالات متعددة حيث يخصص لكل كلمة معنى ووظيفة واحدة، **مثال: عبارة** "if (statement)"، لا يمكن أن تأخذ أكثر من معنى باختلاف سياقها وإنما لها وظيفة ومعنى ثابت في اللغة أينما وجدت.

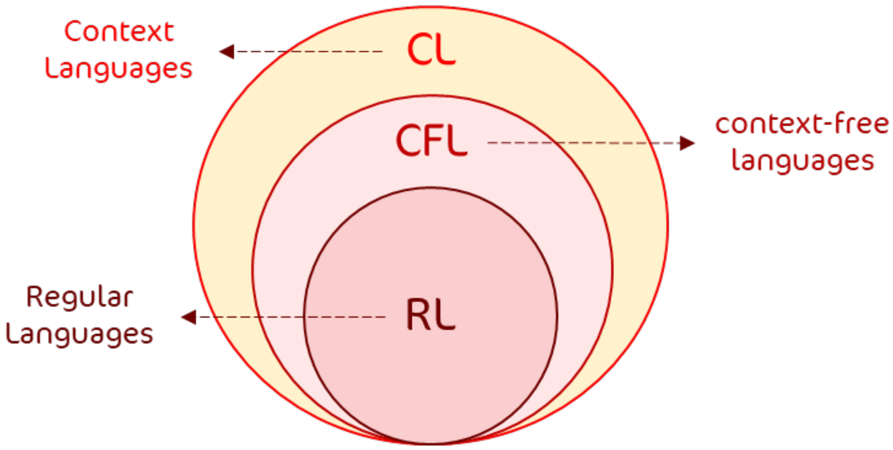
كيف يمكننا التعرف على لغة معينة (Language Recognition)؟

كما ذكرنا سابقاً بأن كل أتومات يقوم بتعريف لغة معينة, وبالتالي يمكننا التعرف على لغة معينة عن طريق الأتومات الذي يقوم بقبولها, أي:

- اللغات المنتظمة (Regular Languages) يمكن التعرف عليها من خلال الأتومات المنتهي (Finite Automata) ومن خلال (Regular Grammar) والذي سنتعرف عليه لاحقاً.
- لغات خارج السياق يمكن التعرف عليها من Push-down Automata أو من خلال Context-free grammar
- اللغات السياقية يمكن التعرف عليها من خلال Turing Machines أو من خلال context grammar.

سنتعرف على كل نوع من الـgrammar في المحاضرات القادمة.

هرمية Chomsky للغات (Chomsky Languages Hierarchy)



حيث أن RL هي جزء من CFL والتي بدورها هي جزء من CL.

بعض التعاريف والمصطلحات

1) الأبجدية (Alphabet): رمزها ($\sum$), وهي مجموعة منتهية من الرموز (symbols) الغير قابلة للتجزئة, أمثلة:

- أبجدية النظام الثنائي هي: $\sum = \{0,1\}$
- أبجدية النظام الثلاثي هي: $\sum = \{0,1,2\}$
- أبجدية النظام العشري هي: $\sum = \{0,1,2,..,8,9\}$
- أبجدية اللغة العربية هي: $\sum = \{$أ, ب, ت, … ….., و, ي$\}$
- أبجدية اللغة الإنكليزية هي: $\sum = \{a, b, c, \ldots\ldots, y, z\}$

- إن المجموعة التالية: {01,1} لا تمثل أبجدية لأن 01 هو قابل للتجزئة أي أنه ليس رمز وإنما سلسلة من رموز.
- المجموعة الغير منتهية من الرموز لا تشكل أبجدية أيضاً فمثلاً مجموعة الأعداد الطبيعية N لا تشكل أبجدية.

2) الكلمة (string / word): هي سلسلة من رموز الأبجدية($\sum$).

3) ε (Epsilon): كلمة مميزة خالية لا تحتوي على أي رمز ومحتواه في كل اللغات.

مثال:

بفرض لدينا الأبجدية التالية: $\Sigma = \{0,1\}$

فتشكل كل من السلاسل الآتية كلمات:

$$01,001,111,101010,00001111, \ldots.$$

4) طول الكلمة (Length word): نعرف طول الكلمة (w) بأنه عدد الرموز المشكلة للكلمة (سواءً كانت مكررة أو لا) ونرمز له بالرمز : $|w|$

مثال:

$$w_1 = 01 \rightarrow \ |w_1| = 2$$
$$w_2 = 0011 \rightarrow \ |w_2| = 4$$
$$|\varepsilon| = 0$$

5) اللغة (Language): نرمز لها غالباً بالرمز (L), وهي مجموعة كلمات (سلاسل) مبنية على أساس أبجدية معينة, ومن الممكن أن تكون هذه المجموعة منتهية أو لا, وتخضع الكلمات لقواعد محددة.

أمثلة:

- $L1 = \{a, aa, ab\}, based\ on\ the\ alphabet\ \Sigma = \{a, b\}$
- $L2 = \{001,101,111,001\}, based\ on\ the\ alphabet\ \Sigma = \{0,1\}$

في المثالين السابقين نلاحظ أن كلا من $L1$ & $L2$ هي عبارة عن مجموعة منتهية من الكلمات.

في حال كانت اللغة تحتوي على مجموعة غير منتهية من الكلمات يمكننا التعبير عن اللغة بالخاصة التي تميز كلماتها كما في المثال التالي:

- بفرض لدينا اللغة $L3$ *مبنية على الأبجدية* $\{0,1\}$ وجميع كلماتها تنتهي بالسلسلة 01 فنعرف اللغة كما يلي:

$$L3 = \{w : \ ends\ with\ 01\ \}, based\ on\ the\ alphabet\ \ \Sigma = \{0,1\}$$

نلاحظ أنه يوجد عدد لا نهائي من الكلمات التي تنتمي لهذه اللغة مثل :

$$01,001,0101,001101,1111101, \ldots \ldots \ldots.$$

ملاحظة هامة: اللغة هي مجموعة من الكلمات المتشكلة من أبجدية وليست كل الكلمات التي يمكن تشكيلها من الأبجدية, مثال في اللغة العربية تعتبر الكلمة "أأأت" كلمة متشكلة من الأبجدية العربية ولكنها لا تنتمي إلى اللغة العربية لأنها كلمة غير صالحة وليس لها معنى.

العمليات على اللغة

بما أن اللغة هي مجموعة, فجميع العمليات التي يمكننا تطبيقها على المجموعات يمكننا تطبيقها على اللغات.

التقاطع ($\cap$):

$$L1 \cap L2 = \{w : w \in L1 \ \wedge \ w \ \in L2\}$$

أي أنها مجموعة الكلمات w التي تنتمي إلى **كلا** اللغتين معاً.

الاجتماع (∪):

$$L1 \cup L2 = \{w : w \in L1 \ \vee \ w \ \in L2\}$$

أي أنها مجموعة الكلمات w التي تنتمي إلى **أحد** اللغتين أو **كليهما**.

الفرق (\):

$$L1 \setminus L2 = \{w : w \in L1 \ \wedge \ w \ \notin L2\}$$

أي أنها مجموعة الكلمات w التي **تنتمي** إلى اللغة الأولى و**لا تنتمي** إلى اللغة الثانية .

الإغلاق "Kleene closure" (*):

بدايةً وقبل تعريف هذه العملية على اللغات سنعرف قوى الأبجدية:بفرض لدينا الأبجدية $\sum = \{a, b\}$:

- Σ^0: هي مجموعة الكلمات ذات الطول صفر أي السلاسل الخالية (بشكل افتراضي هي عبارة عن ε) أي: $\sum^0 = \{\varepsilon\}$
- Σ^1: هي مجموعة الكلمات ذات الطول واحد (محرف واحد) وفي مثالنا ستكون: $\sum^1 = \{a, b\}$
- Σ^2: هي مجموعة الكلمات ذات محرفين وفي مثالنا ستكون: $\sum^2 = \{aa, ab, ba, bb\}$
- Σ^k: هي مجموعة الكلمات ذات الطول k.

نعرّف عملية Kleene closure (Σ^*) على أبجدية بأنها مجموعة <u>كل الكلمات من كل الأطوال</u> التي يمكن **تشكيلها** من الأبجدية Σ أي أنه عبارة اجتماع Σ^0و Σ^1 و Σ^2 و Σ^3 وهكذا ...

$$\Sigma^* = \sum^0 \cup \sum^1 \cup \sum^2 \ \cup \ ...$$

وبشكل مماثل نعرف العملية السابقة (Kleene closure) <u>للغة</u> L كالتالي:

$$L^* = L^0 \cup L^1 \cup L^2 \ \cup ...\ ...\ ...\ ....$$

حيث بفرض $L = \{a, ab\}$:

فيكون L^2 هي مجموعة من تشكيل L مع L أي ستكون **تشكيل جديد من الكلمات** بحيث كل كلمة جديدة تتشكل من دمج concatenation لكلمتين من اللغة:

$$L^2 = \{aa, aab, aba, abab\}$$

وهكذا بالنسبة لL^3 و L^4 و ...

الوصل "concatenation" (.)

بفرض لدينا كلمة أولى $w1 = 01$ وكلمة ثانية $w2 = 10$ فتشكل عملية الوصل كلمة جديدة كالتالي:

$$w1.w2 = 0110$$

وتكون الكلمة ε عنصر حيادي بالنسبة لعملية وصل الكلمات أي أن:

$$\varepsilon.w \ = \ w.\varepsilon \ = \ w$$

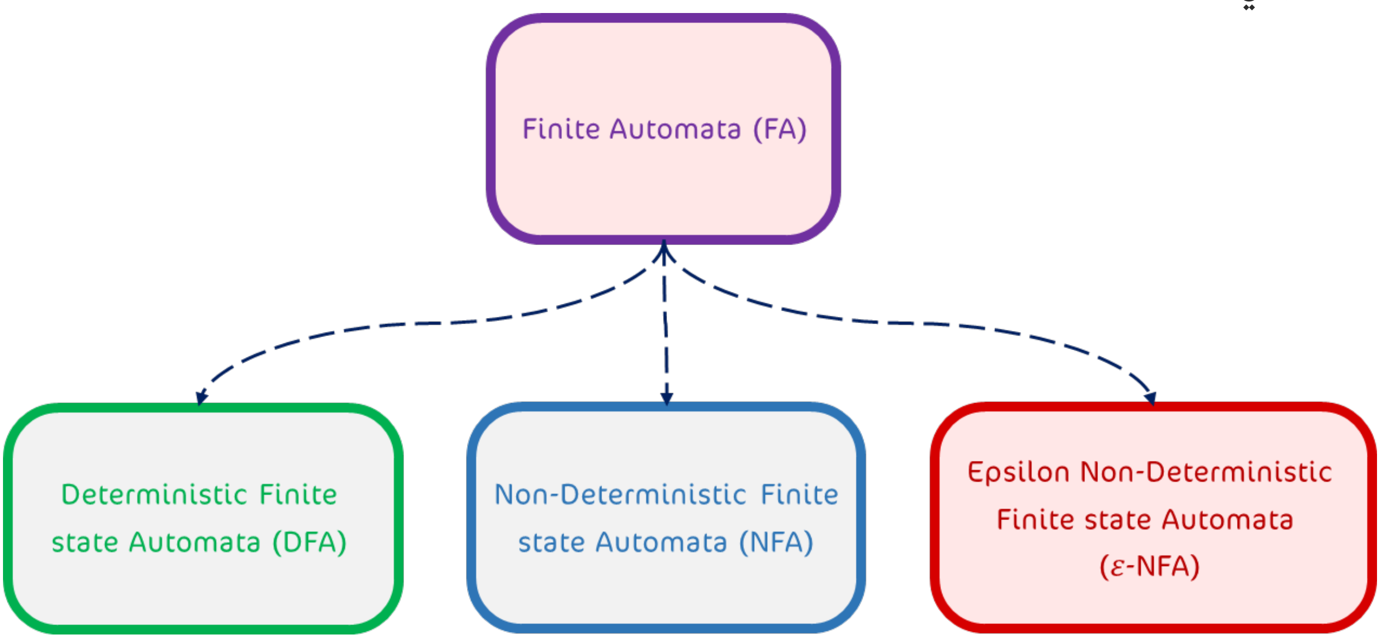
ملاحظات

1. إن Σ تختلف عن Σ^1 بحيث أن الأولى هي الأبجدية التي تتكون من الرموز التي هي أصغر وحدة بناء بينما الثانية هي عبارة عن سلاسل من الطول واحد وبالتالي الاختلاف بينهما يكون مثل الاختلاف بين $'a'$ و $"a"$ حيث الأولى هي محرف (رمز) والثانية هي سلسلة من الطول واحد.
2. بما أن Σ^* تحتوي على جميع الكلمات الممكن تشكيلها باستخدام أبجدية معينة فهذا يعني أن اللغة L هي محتواه في Σ^* أي أن: $L \subseteq \Sigma^*$
3. نعرف المجموعة المتممة للغة L والتي نرمز لها ب L' حيث : $L' = \Sigma^* \backslash L$
4. (كما ذكرنا سابقاً) اللغة لا تتكون من كل الكلمات الممكن تشكيلها من أبجدية وإنما تتكون اللغة من مجموعة منها فقط, لكن هنالك لغات قد تحتوي على جميع التشكيلات من الرموز الممكن تكوينها

مثال: بفرض لدينا الأبجدية الثنائية: $\Sigma = \{0,1\}$
إن أي تشكيل لكلمة من هذه الأبجدية مهما كان طولها سينتمي إلى لغة الأرقام الثنائية الصالحة لأن أي تركيب من الأبجدية السابقة سيولد رقم ثنائي له معنى.

الأتومات المنتهي finite automata (FA)

ستتناول الآن النوع الأبسط من الأتومات والذي هو الأتومات المنتهي (finite automata), يوجد ثلاثة أنواع من هذا الأتومات وهي :



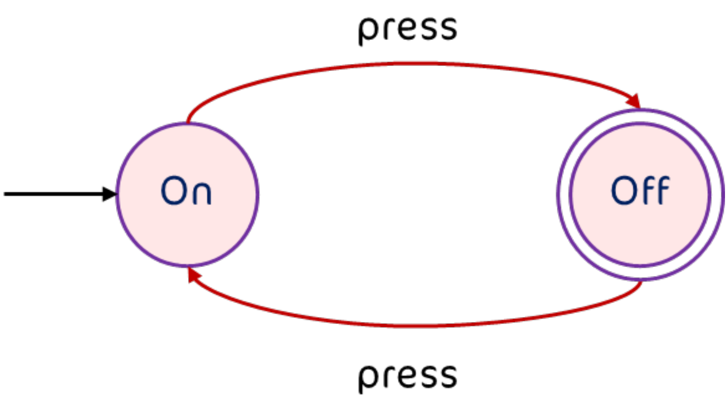
تتميز جميع الأنواع السابقة من الأتومات بأن الأتومات يمر بحالات منتهية (finite) ويتم الانتقال من حالة إلى أخرى بحدث معين أو إدخال معين.

أمثلة عامة عن ال FA

مثال 1:

يمكننا تمثيل جهاز القاطعة (switch) على شكل **أتومات منتهي** يمر بحالتين: off || on ويتم الانتقال بين الحالتين السابقتين بحدث الضغط (pressing action), ونرسم **تصميم** لهذا الأتومات على الشكل التالي:

- الحالة التي يوجد سهم داخل إليها من الخارج تمثل **الحالة الابتدائية** (initial state) للأتومات (وهنا قمنا اختيارياً بجعل الحالة on هي الحالة الابتدائية).
- الحالة التي يوجد دائرة تحيط بها من الخارج تمثل **حالة نهائية** (final state) وسنوضح معنى الحالة النهائية للأتومات في أمثلة لاحقة.



بشكل بسيط يمكننا أن نفسر دلالة الحالة النهائية كالتالي:
بفرض لدينا كلمة (والتي هي عبارة عن سلسلة من الرموز) وتم إدخالها على أتومات وعند انتهاء الكلمة كانت حالة الأتومات هي حالة نهائية فتكون هذه الكلمة مقبولة في هذا الأتومات وتنتمي إلى اللغة الذي يعرّفها, وعموماً يمكن أن يكون للأتومات مجموعة من الحالات النهائية وليس بالضرورة أن تكون هنالك حالة نهائية وحيدة وذلك يعتمد على وظيفة الأتومات.

مثال 2:

من الاستخدامات الهامة للأتومات تكمن في ال compiler theory .

حيث أن ال compiler هو نظام خاص مهمته تحويل نص برمجي من لغة مصدر (source code) عالية المستوى إلى لغة أخرى هي destination code وتكون هذه اللغة أبسط منها.

يمر نظام الcompiler بعدة مراحل سنذكر منها المرحلة الأولى والتي هي التحليل المفرداتي (lexical analyzer) حيث يتم في هذه المرحلة تحويل ال source code إلى مجموعة من ال tokens (أي مجموعة من الكلمات) وفي هذه المرحلة يتم استخدام الأتومات في عدة نواحي:

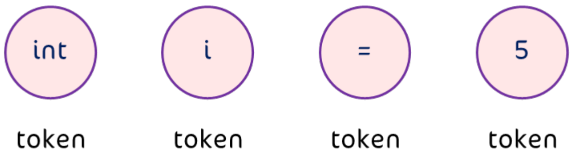
- التعرف على الكلمات المفتاحية (recognizing keywords).
- قبول / رفض تسمية المتحولات (accepting variables naming).

فمثلاً:

ليكن لدينا النص البرمجي التالي :

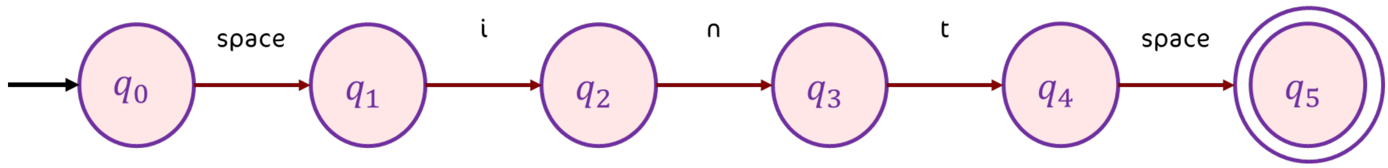
$$int\ \ i = 5;$$

سيتم بداية تحويل النص السابق إلى مجموعة من ال tokens كما يلي :



ثم سيتم تحديد الكلمات المفتاحية من العبارة السابقة باستخدام مجموعة من الأتومات المخصصة للكلمات المفتاحية (لكل كلمة مفتاحية يوجد أتومات خاص بها).

1. الأتومات المخصص للكلمة المفتاحية int:



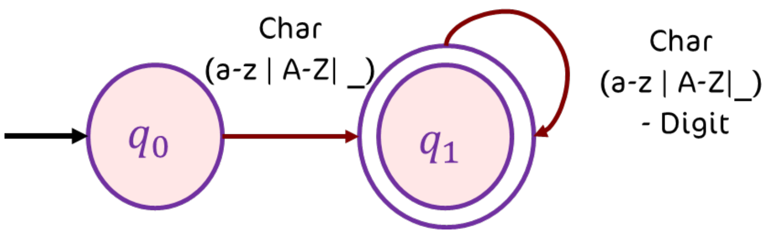
نلاحظ أنه في الأتومات السابق تكون نبدأ من حالة ابتدائية q0 وننتقل إلى الحالة النهائية final state بعد السلسلة التالية من الإدخالات:

$$space -> i -> n -> t -> space$$

وعندما تصل سلسلة الإدخالات إلى الحالة النهائية يقبل هذا الأتومات هذه الكلمة ويتعرف عليها على أنها الكلمة مفتاحية (والتي في مثالنا هي int).

2. الأتومات المخصص لقبول تسمية المتحول:

يتم تحديد فيما إذا كانت تسمية المتحول مقبولة أو لا عن طريق الأتومات التالي:



نلاحظ أنه في الأتومات السابق تكون الحالة الابتدائية عند حالة ال space تم إدخالها سابقاً وعند إدخال أي قيمة char أو _ ينتقل الأتومات إلى الحالة النهائية ويبقى في الحالة النهائية في حال كانت بقية الإدخالات هي char أو digit أو _ وعند انتهاء سلسلة الإدخالات الخاصة بتسمية المتحول عند حالة نهائية للأتومات يتم قبول تسمية المتحول بالشكل المدخل.

Deterministic Finite state Automata (DFA)
الأتومات المنتهي الحتمي

يتميز هذا النوع من الأتومات بما يلي:

- الانتقال لحالة معينة عند دخل محدد هو انتقال وحيد (لا يمكن الانتقال من حالة معينة إلى حالتين مختلفتين عند دخل محدد).
- من أجل كل حالة من حالات الأتومات يجب تعريف انتقالات منها حسب كل دخل ممكن (أي حسب كل رمز من رموز الأبجدية).

مثال:

بفرض الأبجدية هي $\{0,1\}$ والحالات هي q_1, q_2 فيجب تعريف انتقال من q_0 حسب كلا الدخلين 1 و 0 وكذلك بالنسبة للحالة q_1 فيجب تعريف انتقال حسب كلا الدخلين 1 و 0 ولا يمكن الاكتفاء بتحديد الانتقال حسب أحد الدخلين فقط بل يجب ذكر انتقال الحالات حسب كل رمز من رموز الأبجدية).

- لا توجد حالة الانتقال من حالة إلى أخرى عن طريق ε (أي حالة عدم وجود دخل أساساً).
- كلمة finite تدل على أن عدد الحالات التي يمر بها الأتومات منتهية وبالتالي نحتاج إلى ذاكرة محددة وكلمة deterministic تدل على الانتقال الحتمي و المحدد بين الحالات كما بيّنّا سابقاً.

التعريف الرياضي لأتومات ال DFA:

يمكن تعريف الأتومات المنتهي الحتمي رياضياً بالخماسية التالية:

$$(Q\ ,\Sigma\ ,q_0\ ,\delta, F)$$

حيث :

Q : هي مجموعة الحالات التي يمر بها الأتومات (set of automata's states).

Σ : هي أبجدية اللغة (alphabet).

q_0 : هي الحالة الابتدائية للأتومات (initial state).

δ : هو تابع الانتقال بين الحالات (transition function) والذي يعرف رياضياً بالشكل الآتي:

$$\delta : Q \times \Sigma \rightarrow Q$$

أي أنه يربط حالة محددة من Q عند إدخال رمز من Σ إلى حالة وحيدة من Q.

F : هي مجموعة الحالات النهائية من Q (set of final states).

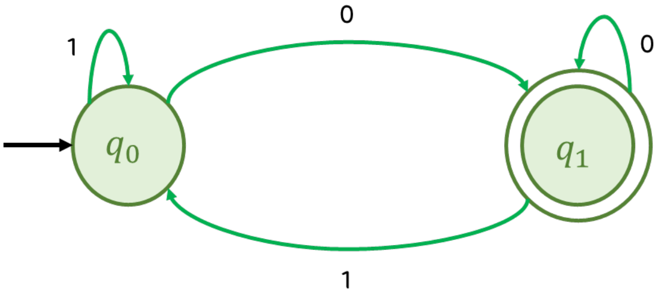
- ذكرنا سابقاً أن كل أتومات يعرف لغة خاصة به, فما هي اللغة التي يعرفها أتومات معين؟
إن اللغة التي يعرفها أتومات معين هي جميع الكلمات التي يقبلها هذا الأتومات
- وهنا نسأل: ما هي الكلمات التي يقبلها الأتومات؟
إن الكلمات التي يقبلها الأتومات هي الكلمات التي تكون بدايةً في الحالة البدائية للأتومات وتنتهي بأحد الحالات النهائية من حالات الأتومات النهائية.

تمارين :

تمرين 1 :

ليكن لدينا ال DFA التالي:

ما هي الخماسية الرياضية الخاصة بهذا الأتومات؟ وما اللغة التي يعرفها هذا الأتومات؟



الحل :

مجموعة الحالات في هذا الأتومات هي: $Q \; = \{\, q_0, q_1\}$

أبجدية هذا الأتومات هي: $\Sigma \; = \; \{0,1\}$

الحالة الابتدائية للأتومات هي الحالة: q_0

الحالات النهائية لهذا الأتومات هي الحالة q_1 فقط, أي : $F \; = \{\, q_1\}$.

يمكننا التعبير عن تابع انتقال الحالات δ عن طريق جدول الانتقالات كما يلي:

	0	1
q_0	q_1	q_0
q_1	q_1	q_0

نلاحظ من خلال الأتومات السابق أنه بمجرد أن يكون الدخل 0 ننتقل إلى الحالة النهائية للأتومات وإذا كان الدخل 1 نعود إلى حالة ليست بحالة نهائية, فيمكننا الاستنتاج بأن الهدف من هذا الأتومات هو أن لا تنتهي سلسلة الدخل عند المحرف 1, أي أنه أي سلسلة لا تنتهي بالمحرف 1 (أي تنتهي بالمحرف 0) ستصل إلى الحالة النهائية وبالتالي سيقبل الأتومات هذه السلسلة (الكلمة) وبالتالي يمكننا استنتاج أن هذا الأتومات يعرف لغة والتي هي جميع الكلمات التي تنتهي بالمحرف 0 ونلاحظ بأن كلمات هذه اللغة عددها لا نهائي وبالتالي اللغة غير محدودة.

أمثلة توضيحية لعدة كلمات ندخلها على هذا الأتومات:

- لتكن لدينا السلسلة (الكلمة) التالية:

$$0011$$

نبدأ بأخذ محرف محرف من اليسار إلى اليمين ونتابع مع الأتومات السابق كما يلي:

بداية نحن عند الحالة الابتدائية q_0 نأخذ المحرف الأول 0 فننتقل إلى الحالة q_1 ثم نأخذ المحرف التالي 0 فنبقى عند الحالة q_1 ثم نأخذ المحرف التالي 1 فننتقل إلى الحالة q_0 ونصل إلى المحرف النهائي 1 ونبقى عند الحالة q_0 , بما أن الحالة التي انتهت الكلمة عندها هي حالة ليست حالة نهائية final state فالكلمة التي أُدخلت على الأتومات مرفوضة ولا تنتمي إلى اللغة التي يعرفها الأتومات.

- لتكن لدينا السلسلة (الكلمة) التالية:

110

بداية نحن عند الحالة q_0 نأخذ المحرف الأول 1 فنبقى عند الحالة q_0 ثم نأخذ المحرف التالي 1 ونبقى عند الحالة q_0 ثم نأخذ المحرف الأخير 0 فننتقل إلى الحالة q_1 والتي هي حالة نهائية وبالتالي يقبل الأتومات هذه الكلمة وستنتمي هذه الكلمة إلى اللغة التي يعرفها الأتومات.

تمرين 2 :

صمم DFA بحيث يقبل 01 ضمن الكلمات المقبولة.

الحل:

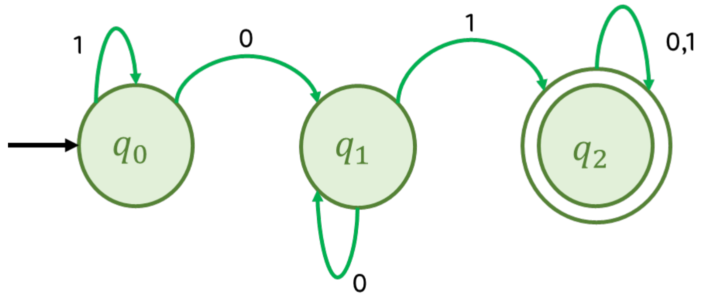
- أولاً يجب علينا أن نفهم بشكل جيد ما هي الوظيفة المطلوبة من هذا الأتومات, نلاحظ هنا أن المطلوب من الأتومات هو قبول أي كلمة تحتوي بداخلها على التسلسل التالي "01" .
- لنصمم هذا الأتومات علينا أن نفكر ما هي الحالات التي يمر بها هذا الأتومات, ولحصر عدد هذه الحالات سنتبع استراتيجية التفكير بالحالات التي يمر بها الأتومات عند تمرير سلسلة مقبولة إليه (ولتكن هي الكلمة المقبولة 001 مثلاً) حيث أنه عند إدخال كل محرف سوف يقربني من الحالة المقبولة, سنعرف حالة جديدة في الأتومات ومن ثم سنستنتج الانتقالات بين الحالات عند كل رمز من رموز الأبجدية, لنوضح الكلام السابق كالتالي:
- بداية سيبدأ الأتومات بحالة ابتدائية ولتكن q_0 وسنبدأ بأول محرف من السلسة المقبولة التي اخترناها والذي هو 0 فننتقل إلى حالة جديدة هي q_1 (لماذا انتقلنا إلى حالة جديدة ؟ لأننا اقتربنا من الوصول إلى حالة قبول السلسلة بمقدار خطوة) ثم سنأخذ المحرف التالي 0 ونلاحظ بأن هذا المحرف لن يقربني من الحالة النهائية أو ينقلني إليها فلن نعرف حالة جديدة هنا وسنبقى في الحالة السابقة والتي هي q_1 .

- لماذا بقينا في q_1 ولم ننتقل إلى q_0 مثلاً ؟
  لأنه من q_0 نحتاج إلى دخلين 0 ثم 1 لنصل إلى الحالة النهائية ولكن في حالتنا هذه نحتاج إلى الدخل 1 فقط للوصول إلى الحالة النهائية والحالة التي تمثل هذا هي الحالة q_1 ثم سنأخذ المحرف 1 وبالتالي تحقق هدف الأتومات وحصلنا على التسلسل "01" في الكلمة ووصلنا إلى الحالة النهائية التي هي q_2 .

إلى الآن حصرنا عدد الحالات وبعض الانتقالات الخاصة ولكن لم نكمل بقية الانتقالات بين الحالات:

- في حال كنا في الحالة q_0 وتلقينا الدخل 1 فإننا سنبقى في الحالة نفسها لأننا لم نقترب من الوصول إلى الحالة النهائية.
- في حال كنا في الحالة النهائية q_2 وتلقينا أحد الدخلين 0 أو 1 فسنبقى في الحالة النهائية q_2 لأنه لا يهم ما يأتي في سلسلة المحارف بعد أن تحقق ورود التسلسل "01" مرة واحدة على الأقل ولن تتغير لدينا الحالة.
- وذكرنا في الملاحظة الانتقال في الحالة q_1 عند ورود الدخل 0 .

وهكذا نكون ذكرنا جميع حالات الانتقال بين حالات الأتومات فنحصل على الأتومات DFA التالي:

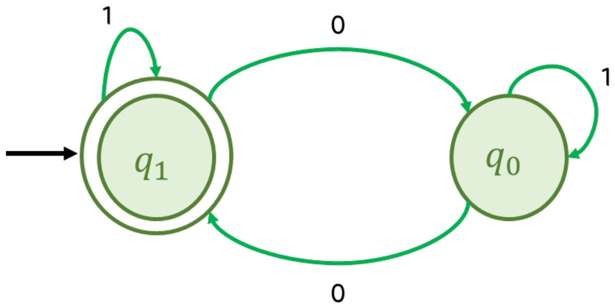


تمرين 3:

صمم DFA تقبل جميع الأعداد الثنائية التي تحتوي على عدد زوجي من الأصفار .

الحل:

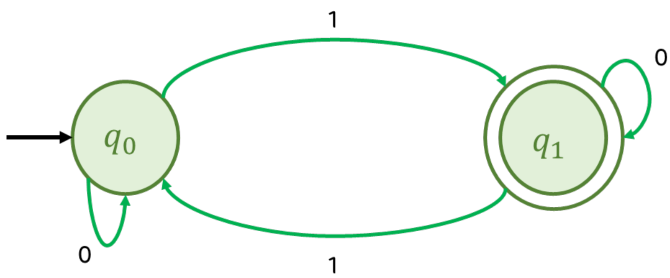
- يمكننا ملاحظة أنه بأي عدد ثنائي لدينا حالتين فقط:
حالة عدد زوجي من الأصفار (والتي تضم حالة عدم ورود أي صفر في الرقم)
وحالة عدد فردي من الأصفار.
- في الحالة الابتدائية لن يكون هنالك أي دخل وهذا يقتضي أنه يوجد عدد قدره 0 من الأصفار وهو عدد زوجي فستنتج أن الحالة الابتدائية للأتومات هي حالة نهائية وسنسميها q_0 وسنبقى بهذه الحالة طالما الدخل هو 1.
- وفي حال تم إدخال 0 ننتقل إلى حالة جديدة q_1 وهي حالة عدد فردي من الأصفار ونبقى بهذه الحالة طالما أن الدخل 1, ونرجع إلى الحالة الأولى q_0 عند إدخال أي صفر إضافي لأنه تحول عدد الأصفار إلى عدد زوجي .
- وبالتالي سنحصل على ال DFA الآتي :



تمرين 4:

صمم DFA يقبل الأعداد الثنائية التي تحتوي على عدد فردي من الواحدات.

الحل: بنفس منطق التمرين السابق يمكننا استنتاج DFA التالية :



ملاحظة: في حال كانت الحالة الابتدائية هي حالة نهائية كما في التمرين 3 فنستنتج مباشرةً أن الكلمة ε تنتمي إلى اللغة التي يعرفها هذا الأتومات لأن هذه الكلمة تمثل حالة عدم ورود أي دخل وبالتالي ما زلنا في الحالة الابتدائية التي هي حالة نهائية وبالتالي تصبح ε مقبولة في اللغة التي يعرفها الأتومات.

non-deterministic finite state automata (NFA)
الأتومات المتتهي الغير حتمي

يتميز هذا النوع من الأتومات بما يلي:

- إن الانتقال عند حالة معينة وعند دخل محدد هو انتقال ليس بالضرورة أن يكون وحيد (يمكن الانتقال من حالة معينة إلى حالتين مختلفتين عند دخل محدد).
- من أجل كل حالة من حالات الأتومات ليس بالضرورة تعريف انتقالات منها حسب كل دخل ممكن أي حسب كل رمز من رموز الأبجدية, ويمكن الاكتفاء بتحديد بعض الانتقالات حسب وظيفة الأتومات.
- لا توجد حالة الانتقال من حالة إلى أخرى عن طريق ε (أي حالة عدم وجود دخل أساساً).

التعريف الرياضي للأتومات NFA

يمكن تعريف الأتومات المتتهي الغير حتمي رياضياً بالخماسية التالية :

$$(Q\ ,\Sigma\ , q_0\ , \delta, F)$$

حيث :

Q : هي مجموعة الحالات التي يمر بها الأتومات(set of automata's states).

Σ : هي أبجدية اللغة (alphabet).

q_0 : هي الحالة الابتدائية للأتومات(initial state).

δ : هو تابع الانتقال بين الحالات (transition function) والذي يعرف رياضياً بالشكل الآتي :

$$\delta : Q \times \Sigma \rightarrow P(Q)$$

حيث $P(Q)$ هو مجموعة أجزاء Q .

F : هي مجموعة الحالات النهائية من Q (set of final states).

مثال على مفهوم مجموعة أجزاء مجموعة

- بفرض لدينا المجموعة Q= {q0,q1} فتكون مجموعة أجزاء Q هي: $P(Q) = \{\phi, \{q0\}, \{q1\}, Q\}$
- أو بعبارة أخرى: $P(Q) = \{\phi, \{q0\}, \{q1\}, \{q0, q1\}\}$

ملاحظة: في حال كان لدينا مجموعة عدد عناصرها n عنصر فإن مجموعة أجزاء هذه المجموعة عدد عناصرها 2^n.
وبالتالي نستنتج أن δ يربط حالة محددة من Q عند إدخال رمز من Σ إلى أحد عناصر P(Q).

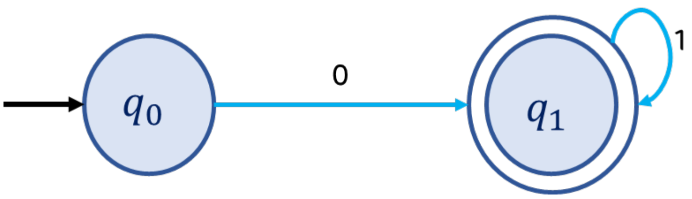
ملاحظات:

1. نلاحظ أن الفرق بالتعريف الرياضي بين DFA & NFA يكمن في تعريف تابع الانتقال δ فقط.
2. نلاحظ أن NFA تشمل ال DFA لأنه نجد أن مستقر تابع الانتقال δ في NFA يشمل مستقر تابع الانتقال δ في DFA والذي هو Q , أي أن :
$Q \subset P(Q)$, وبالتالي كل DFA هو NFA والعكس ليس صحيح بالضرورة.
3. يوجد طريقة لتحويل ال NFA إلى DFA على الرغم من أن ال NFA هي الأشمل وسنتوسع في ذلك في المحاضرات اللاحقة.

تمارين

تمرين 5:

هل ال FA التالي يمثل DFA أم NFA ؟ وفي حال كان يمثل NFA قم بتحويله إلى DFA

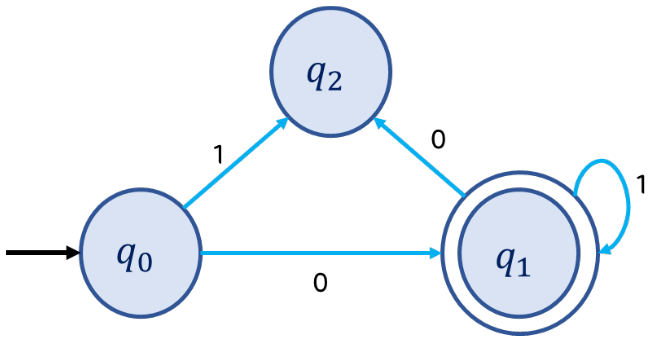


الحل:

- نلاحظ أنه لم يتم تحديد الانتقال من الحالة q_0 عند الدخل 1 , وأيضاً لم يتم تحديد الانتقال من الحالة q_1 عند الدخل 0 وبالتالي لا يمكن أن يكون الأتومات السابق DFA وإنما هو NFA.
- للانتقال من NFA إلى DFA سنقوم بما يلي:

نرسل جميع الانتقالات الغير محددة في ال NFA إلى حالة جديدة ولتكن q_2, ومهما كان الدخل الوارد على الحالة q_2 سنبقى في الحالة ذاتها ولن نستطيع الخروج منها ولذلك تسمى الحالة هذه بالحالة الميتة أو dead state أو trap state.

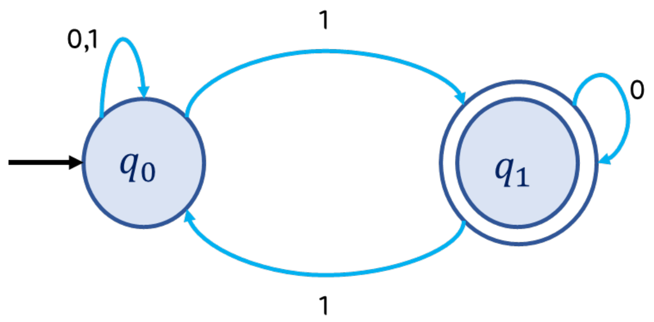
- فنحصل بذلك على ال DFA التالية :



نلاحظ أن اللغة التي يعرفها الأتومات السابق هي مجموعة الكلمات التي تبدأ ب 0 وتُتبع بعدد غير محصور من المحرف 1 أي: 01^*
حيث أن * تعني أنه يمكن أن يتواجد ال 1 بأي عدد كان (أو لا يتواجد من الأساس).

تمرين 6 :

حدد فيما إذا كان ال FA التالي هو DFA أو NFA



نلاحظ أنه فيي الحالة q_0 عند ورود الدخل 1 ننتقل إلى كلا الحالتين q_0 مع الحالة q_1 وبالتالي الانتقال غير وحيد وبالتالي لا يمكن بأن يمثل هذا الأتومات DFA وبالتالي هو NFA.

Epsilon Non-deterministic Finite state Automata (ε-NFA)
الأتومات المنتهي الغير الحتمي مع انتقال ε

يتميز هذا النوع من الأتومات:

- إن الانتقال عند حالة معينة وعند دخل محدد هو انتقال **ليس بالضرورة** أن يكون وحيد (يمكن الانتقال من حالة معينة إلى حالتين مختلفتين عند دخل محدد).
- من أجل **كل حالة** من حالات الأتومات **ليس بالضرورة** تعريف انتقالات منها حسب **كل دخل ممكن** أي حسب كل رمز من رموز الأبجدية, ويمكن الاكتفاء بتحديد بعض الانتقالات حسب وظيفة الأتومات.
- يمكن أن تتواجد حالة الانتقال من حالة إلى أخرى عن طريق ε (أي حالة عدم وجود دخل أساساً).

التعريف الرياضي للأتومات ε-NFA

يمكن تعريف الأتومات المنتهي الغير حتمي من النوع أبسلون رياضياً بالخماسية التالية:

$$(Q\ ,\Sigma\ , q_0\ , \delta, F)$$

حيث :

Q : هي **مجموعة الحالات** التي يمر بها الأتومات(set of automata's states).

Σ : هي **أبجدية** اللغة (alphabet).

q_0 : هي **الحالة الابتدائية** للأتومات(initial state).

δ : هو **تابع الانتقال** بين الحالات (transition function) والذي يعرف رياضياً بالشكل الآتي :

$$\delta : Q\ \times (\ \Sigma\ \cup \varepsilon\ ) \rightarrow P(Q)$$

حيث P(Q) هو مجموعة أجزاء Q .

وهذا يعني أنه يتم ربط حالة من حالات الأتومات Q عند ورود رمز من Σ أو عند عدم ورود أي رمز من الأبجدية (أو بعبارة أخرى عند ورود الدخل ε) إلى عنصر من مجموعة أجزاء المجموعة Q.

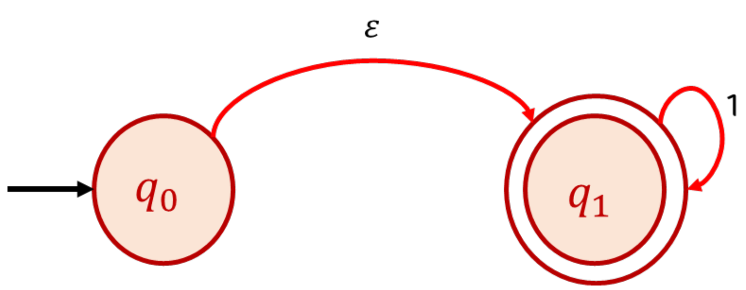
F : هي مجموعة الحالات النهائية من Q (set of final states).

ملاحظات :

1. نلاحظ أن الفرق بالتعريف الرياضي بين ε-NFA & NFA يكمن في تعريف تابع الانتقال δ فقط.
2. نلاحظ أن ε-NFA تشمل ال NFA لأنه نجد أن منطلق تابع الانتقال δ في ε-NFA والذي هو $Q \times (\Sigma \cup \varepsilon)$ يشمل منطلق تابع الانتقال δ في NFA والذي هو $Q \times \Sigma$, أي أن: $\Sigma \subset (\Sigma \cup \varepsilon)$, وبالتالي كل NFA هو ε-NFA والعكس ليس صحيح بالضرورة.
3. بشكل نهائي النوع ε-NFA هو النوع الأشمل من أنواع ال FA.
4. عبارة لا يوجد دخل ⇔ عبارة الدخل هو ε .

تمرين 7 :

حدد نوع ال FA التالي وما هي اللغة التي يعرفها

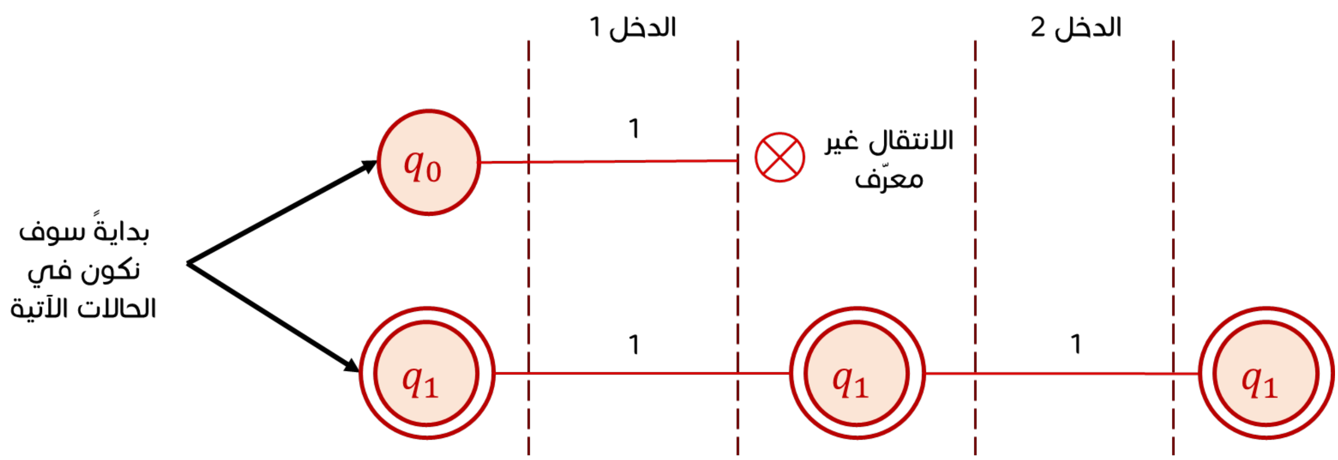


- نلاحظ بأنه يوجد لدينا انتقال من حالة إلى حالة بالدخل ε وبالتالي إن FA هي من النوع ε-NFA .
- يمكننا ملاحظة بأنه في البداية نكون في كلا الحالتين q_0 مع q_1 (نكون في q_0 لأنها الحالة الابتدائية ونكون أيضاً في الحالة q1 لأنه ليس لدينا أي دخل).

أمثلة توضيحية لبعض السلاسل التي سندخلها على هذا الأتومات:

- السلسلة "11":

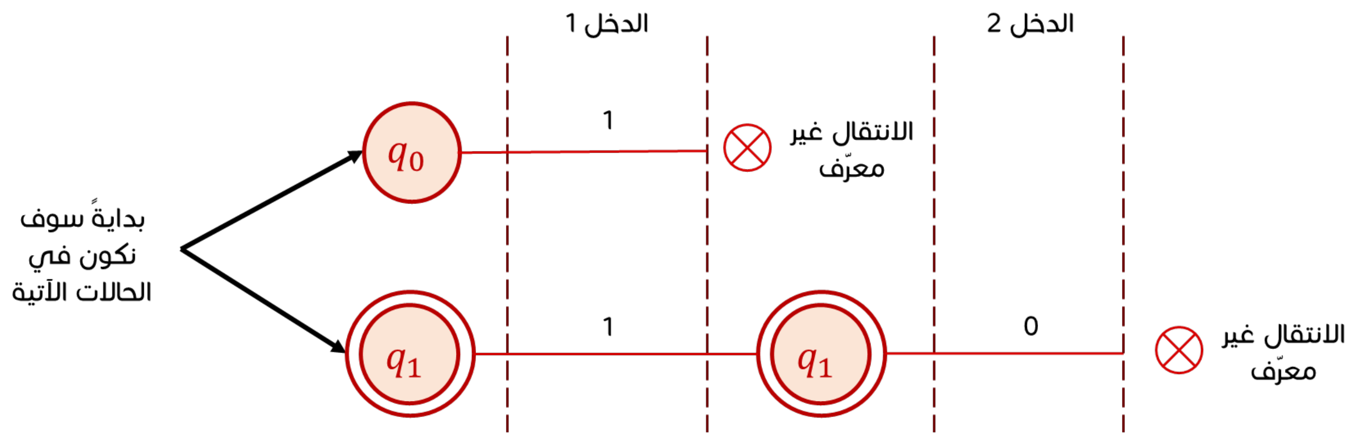
سنقوم بتمثيل الانتقال بين الحالات عن طريق المخطط التالي:



بما أنه انتهينا عند أحد الحالات (التي يمكن أن يكون بها الأتومات) في حالة نهائية والتي هي q_1 فالكلمة **مقبولة وتنتمي** إلى اللغة التي يعرفها الأتومات.

- السلسلة "10":

سنقوم بتمثيل الانتقال بين الحالات عن طريق المخطط التالي:

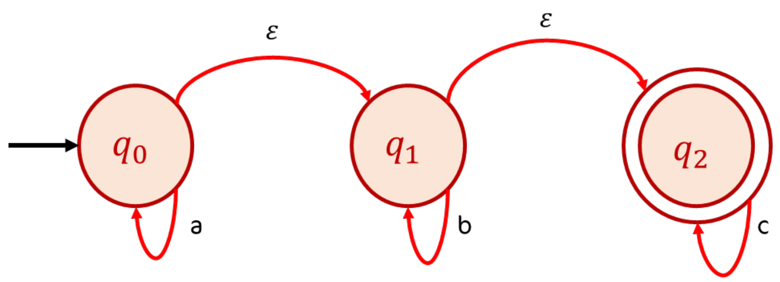


بما أن جميع الحالات (التي يمكن أن يكون بها الأتومات) توقفت في حالة ليست نهائية فالكلمة هي كلمة **غير مقبولة ولا تنتمي** إلى اللغة التي يعرفها الأتومات.

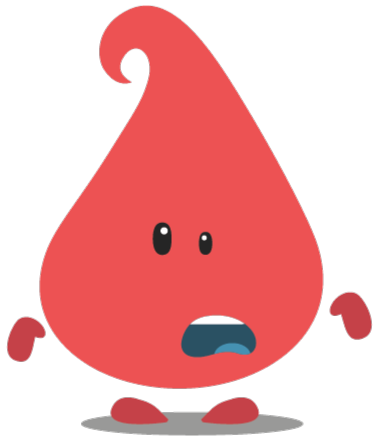
> إن اللغة التي يعرفها هذا الأتومات هي: 1^*
> لأن هذا الأتومات يرفض أي كلمة تحتوي على 0, لأنه لا يوجد أي انتقال صفري معرف في أي حالة من الحالتين (عند عدم وجود انتقال معرف لأحد الرموز الأبجدية تُرفض الكلمة التي تحتوي عليه).

تمرين 8 :

حدد نوع ال FA التالي وما هي اللغة التي يعرفها؟



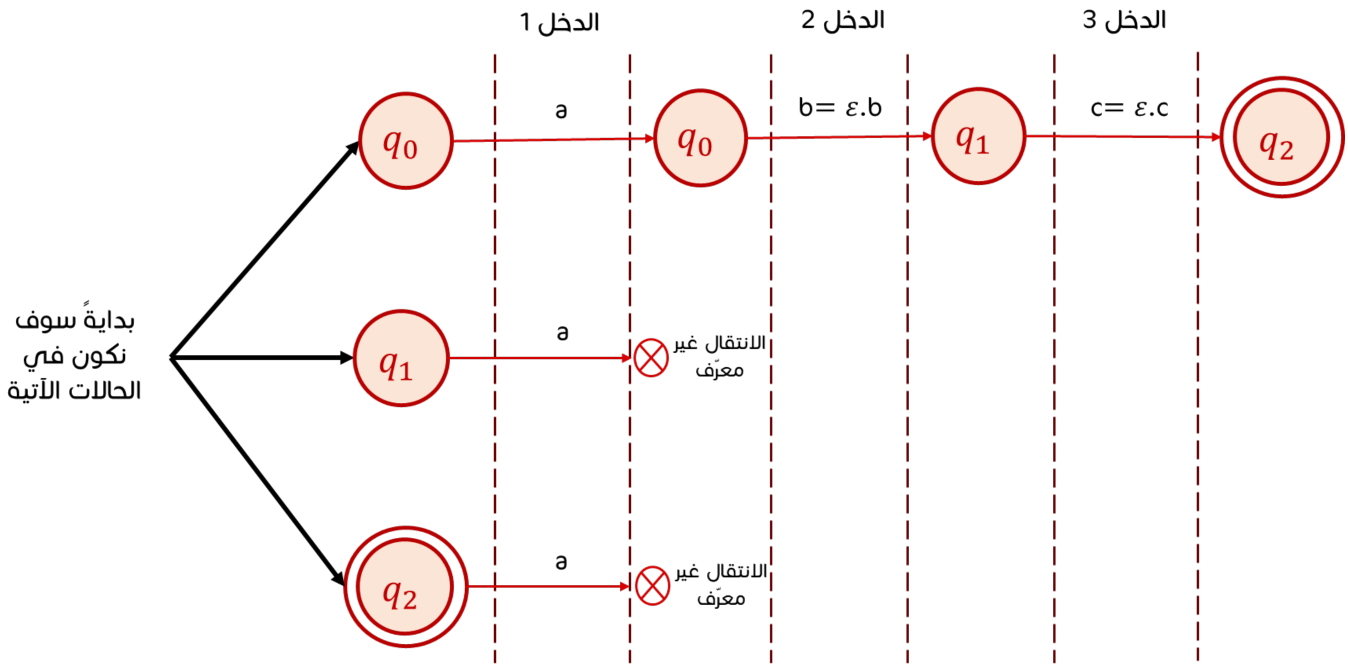
- نلاحظ بأنه يوجد لدينا انتقال من حالة إلى حالة بالدخل ε وبالتالي إن FA هي من النوع ε-NFA .
- يمكننا ملاحظة بأنه في البداية نكون في كلاً من الحالات q_0 مع q_1 مع q_2 (نكون في q_0 لأنها **الحالة الابتدائية** ونكون أيضاً في الحالة q_1 و q_2 لأنه ليس لدينا أي دخل).



أمثلة توضيحية لبعض السلاسل التي سندخلها على هذا الأتومات:

- **السلسلة**"abc":

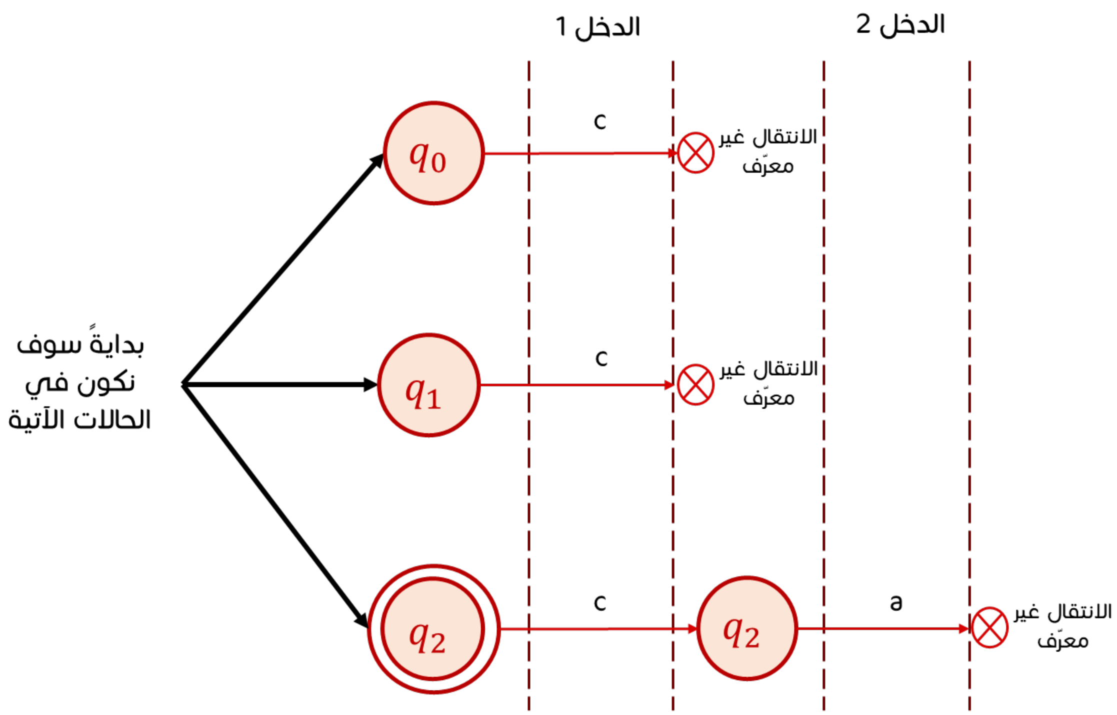
سنقوم بتمثيل الانتقال بين الحالات عن طريق المخطط التالي:



بما أنه انتهينا عند أحد الحالات (التي يمكن أن يكون بها الأتومات) في حالة نهائية فالكلمة هي كلمة مقبولة وتنتمي إلى اللغة التي يعرفها الأتومات.

- **السلسلة**"ca":

سنقوم بتمثيل الانتقال بين الحالات عن طريق المخطط التالي :



بما أنه جميع الحالات (التي يمكن أن يكون بها الأتومات) توقفت في حالة **ليست نهائية** فالكلمة هي كلمة **غير مقبولة** و **لا تنتمي** إلى اللغة التي يعرفها الأتومات.

إن اللغة التي يعرفها هذا الأتومات بنفس منطق التمرين السابق هي: a*b*c*.

ملاحظة:
نقول عن أتوماتين أنهما متكافئين إذا عرّفا اللغة ذاتها
(مثال: الأتوماتان DFA & NFA في التمرين 5 هما أتوماتان متكافئان).

- انتهت المحاضرة -