Model Checking

Amjad Mahfoud

Damascus Univercity

العناوين الرئيسية

1. مقدمة
2. Model checking
3. LTL to NBA
4. كشف التضارب
5. المراجع

مقدمة

مفاهيم عامَّة

مفاهيم عامَّة

النظام التنافسي

- ✓ هي الأنظمة التي يوجد فيها أكثر من إجراء يعملون في نفس الوقت أي هي أنظمة توصف عدة إجراءات تصل إلى أماكن مشتركة للحصول على مصادر منها
- ✓ مشاكله
  - Dead Lock
  - Live Lock
  - Fairness
  - Starvation
- ✓ أي نظام تنافسي يجب أن يحقق الخاصتين التاليتين: Safety، Liveness
- ✓ عملية تشكيل الأنظمة التنافسية وبرمجتها وإكتشاف مشاكلها ليس بالعملية التقليدية أي ليست مجرد تحليل وتصميم بل نحتاج هنا إلى النمذجة

مفاهيم عامَّة

النمذجة

- ✓ هي إستخدام أدوات رياضية لتوصيف المسألة بطرق رياضية (خوارزمية نوعاً ما) والتحقق أن النماذج لا تحوي مشاكل مثل حالات الإقفال الميتة لأنها في الغالب مسائل خطيرة ودقيقة للغاية.
- ✓ بعض الأسئلة المطروحة:
  - ○ كيف نحدد أن النظام ينفذ كما هو متوقع بكل المسارات الموجودة؟
  - ○ هل نستطيع تجريب كل الدخل المحتمل؟
  - ○ هل يمكن معرفة نتائج كل دخل محتمل؟
- ✓ أي نظام تنافسي يجب أن يحقق الخاصتين التاليتين: Safety، Liveness
- ✓ عملية تشكيل الأنظمة التنافسية وبرمجتها وإكتشاف مشاكلها ليس بالعملية التقليدية أي ليست مجرد تحليل وتصميم بل نحتاج هنا إلى النمذجة.

مفاهيم عامَّة

النمذجة

✓ الإختبار و التحقق

الإختبار

- يتم بعد بناء النظام
- يسمح بالتأكد من صحة النتائج للحالات المدروسة فقط.
- لا يسمح بتوقع الحالات التي لن تعمل

التحقق

- تتم على النموذج قبل بناء النظام
- تدرس كل المسارات في النموذج
- من الصعب أن نصل إلى حالة تحقق كاملة ولكن أستطيع أن أقول قدر المستطاع أن نظامي آمن safety
- الإختبار هو حالة خاصة من التحقق

✓ عملية التحقق ضرورية في الأنظمة الأمنية والتجارية والحرجة

مفاهيم عامَّة

الأنظمة الإنتقالية

- ✓ النظام الإنتقالي هو عبارة عن التركيبة التالية

$$\mathbf{M = (\mathit{S}, \rightarrow, \mathit{L})}$$

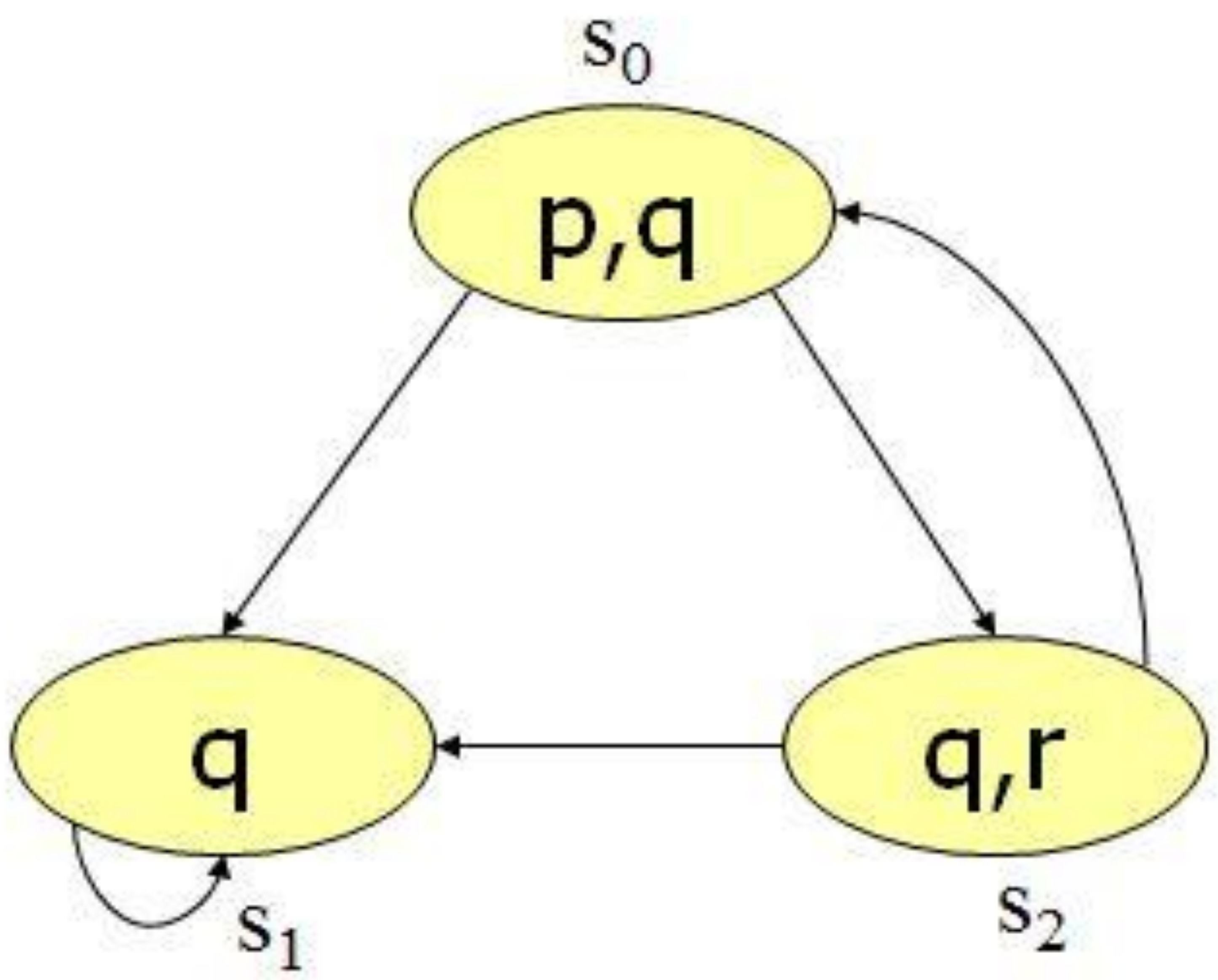
$$\mathbf{L: S \rightarrow P(\ Atoms)}$$

- ✓ المسار هو تسلسل من الحالات ويمثل مستقبل النظام

مفاهيم عامَّة

الأنظمة الإنتقالية

مثال:



$S = \{s_0, s_1, s_2\}$

transitions = $s_0 \rightarrow s_1$, $s_1 \rightarrow s_1$, $s_2 \rightarrow s_1$, $s_2 \rightarrow s_0$, $s_0 \rightarrow s_2$

$L(s_0) = \{p,q\}$

$L(s_1) = \{q\}$

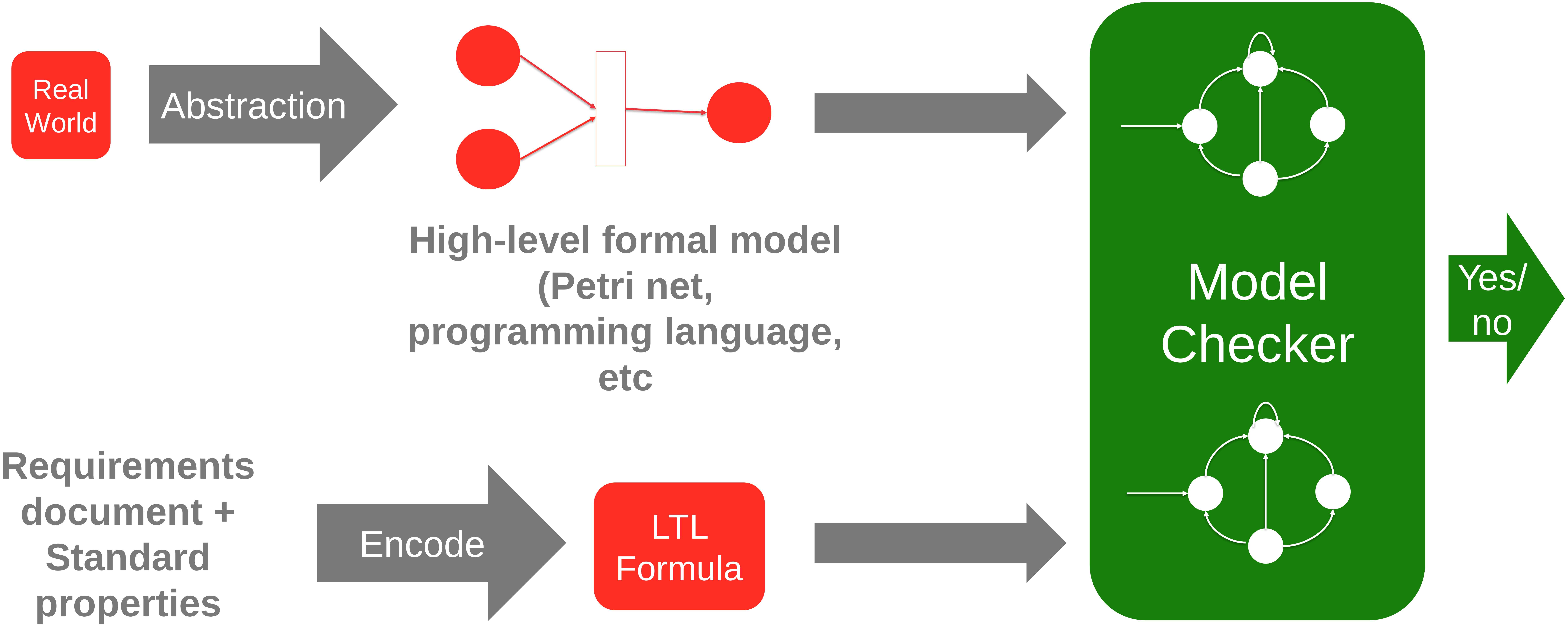
$L(s_2) = \{q,r\}$

Model checking

فحص النموذج

Model checking

مرحلة التحقق



LTL To NBA

LTL To NBA

Linear Temporal logic

القواعد اللغوية تكون على الشكل التالي:

$$\varphi ::= p$$

$$| \ (\sim\varphi) \ | \ (\varphi \wedge \varphi) \ | \ (\varphi \vee \varphi) \ | \ (\varphi \rightarrow \varphi)$$

$$| \ X\varphi \ | \ F\varphi \ | \ G\varphi \ | \ (\varphi U \varphi) \ | \ (\varphi W \varphi) \ | \ (\varphi R \varphi)$$

P : هو متحول يُستخدم حسب المسألة

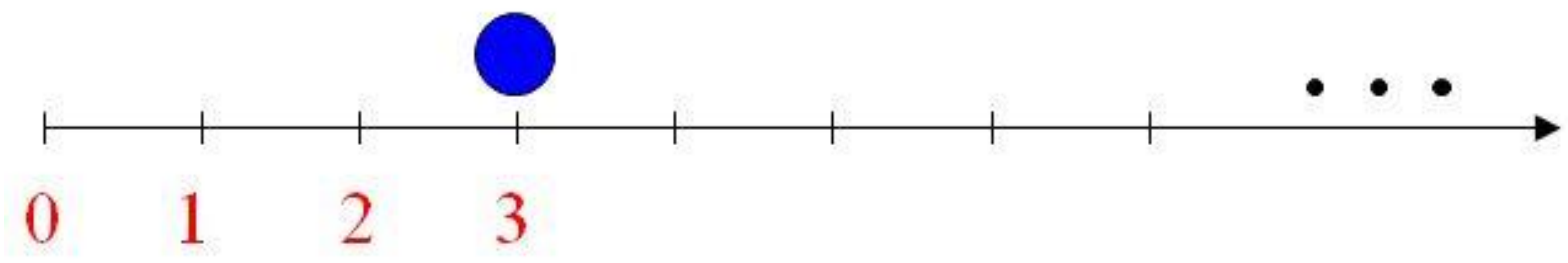
X, F, G, U, R, W: هي علاقات زمنية

LTL To NBA

Linear Temporal logic

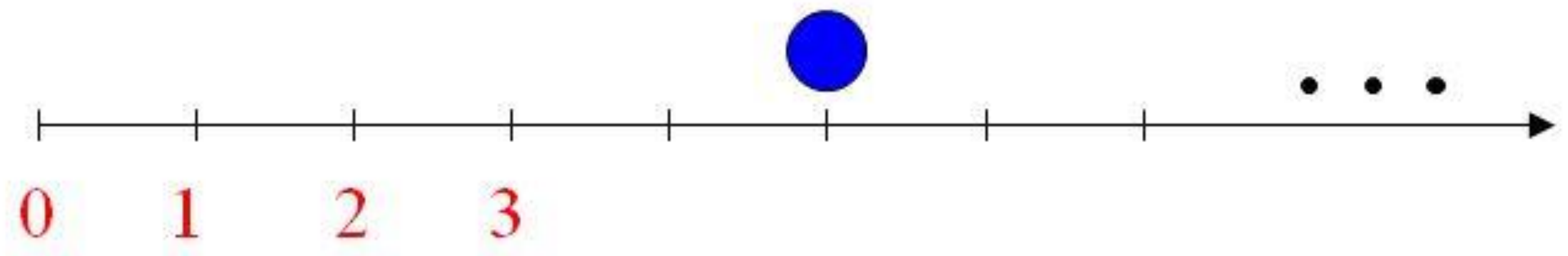
✓ X: تعني الحالة التالية (ne**X**t state)

L(S2) = Xp



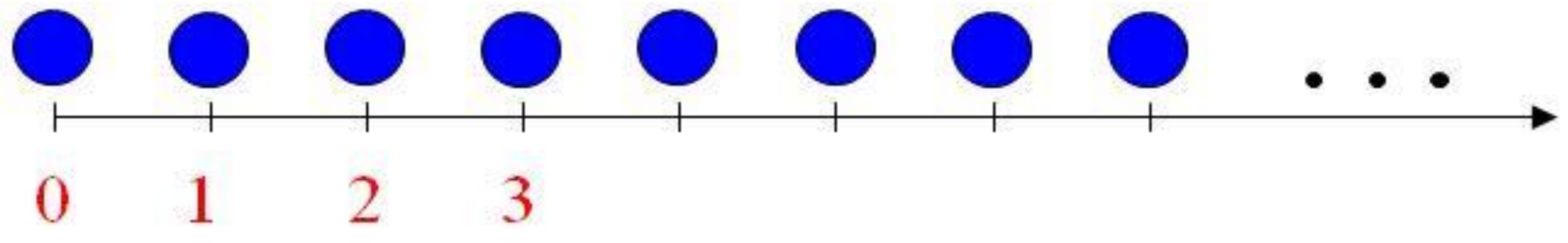
✓ F: تعني في بعض الحالات المستقبلية (some **F**uture state) (Eventually)

L(S0) = Fp



✓ G: تعني كل الحالات المستقبلية (**G**lobally)

L(S0) = Gp

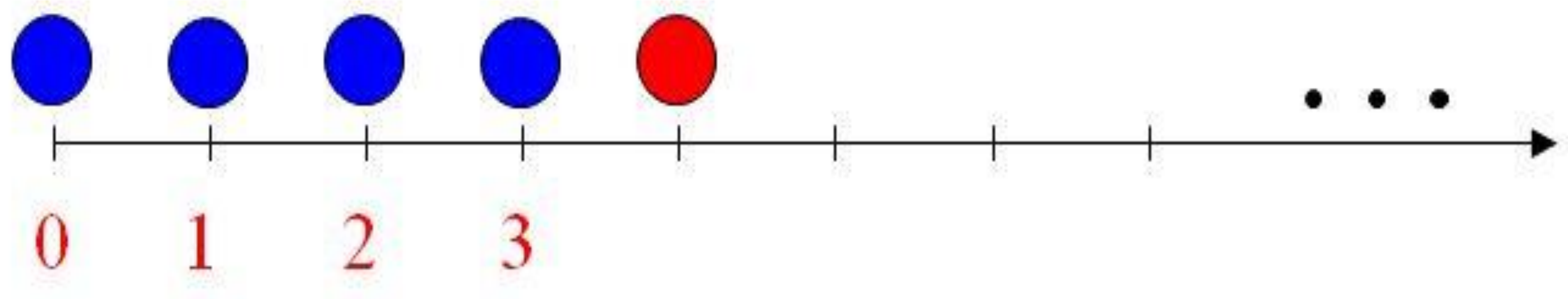


LTL To NBA

Linear Temporal logic

✓ U: (**U**ntil)

L(S0) = p U q



✓ R: (Release)

- **p R q** في حالة معينة s تعني أن q يجب أن تبقى محققة من s وحتى اللحظة التي تصبح فيها p محققة <u>ضمناً</u>
- p R q تكافئ $\neg (\neg p \; U \neg q)$

✓ W : (Weak-Until)

- **p W q** في حالة معينة s تعني أن p يجب أن تبقى محققة من s وحتى اللحظة التي تصبح فيها q محققة أو تبقى p محققة حتى النهاية
- p W q تكافئ $p \, U \, q \vee Gp$

LTL To NBA

Linear Temporal logic

✓ الأوليات هي بالترتيب من الأقوى إلى الأضعف

- ~, X, F, G : العلاقات الأحادية
- /\ : علاقة الـ And
- \/ : علاقة الـ Or
- U, R, W , -> : العلاقات الثنائية

LTL To NBA

أمثلة بسيطة عن عملية التحويل بدون استخدام خوارزمية

لنفرض أنه لدينا:

Atomic Propositions **AP** = { P1, P2 }

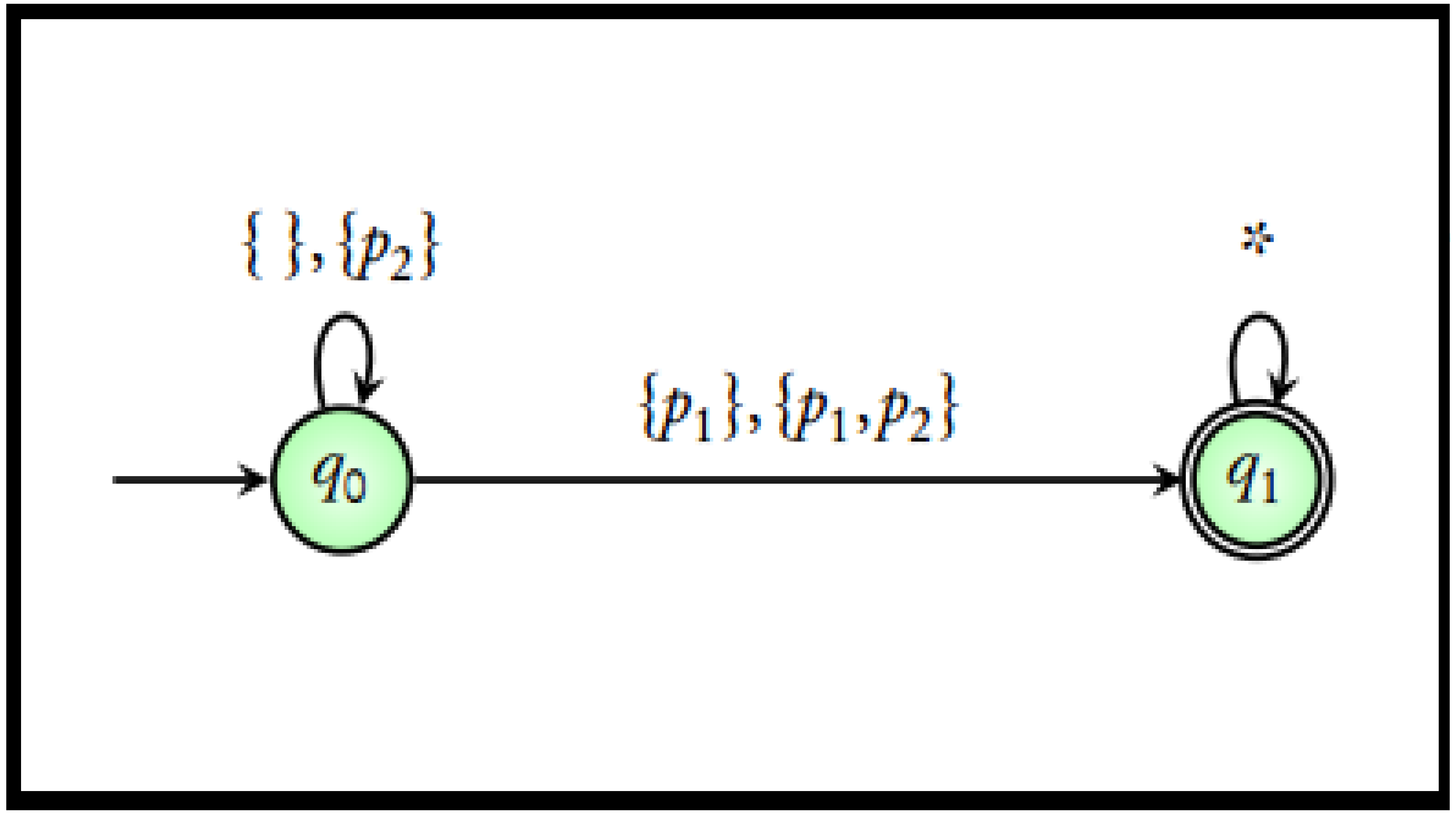
Alphabet = { {}, {P1}, {P2}, {P1, P2} }

LTL To NBA

أمثلة بسيطة عن عملية التحويل بدون استخدام خوارزمية

مثال 1:

F p1

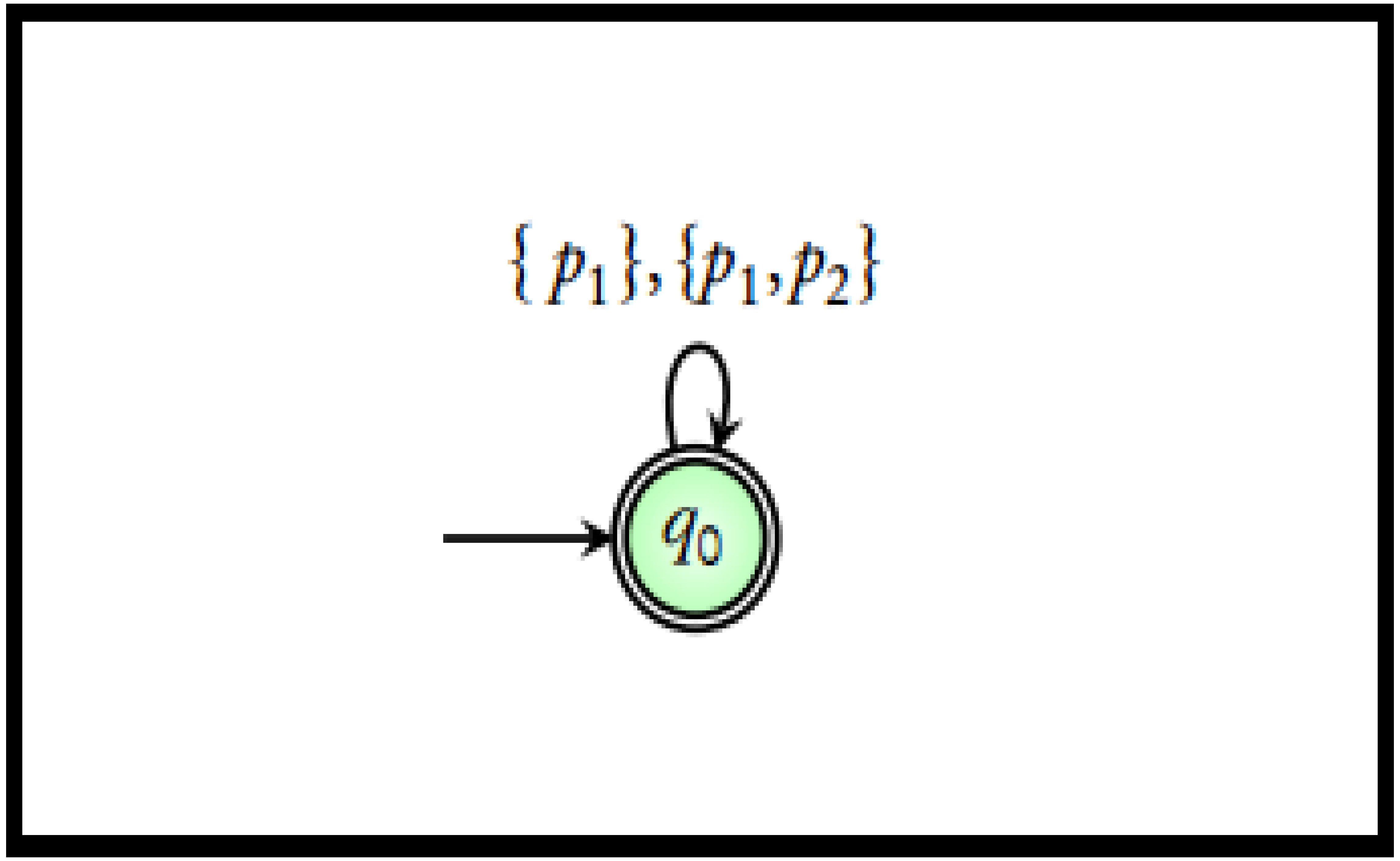


LTL To NBA

أمثلة بسيطة عن عملية التحويل بدون استخدام خوارزمية

مثال 2:

G p1

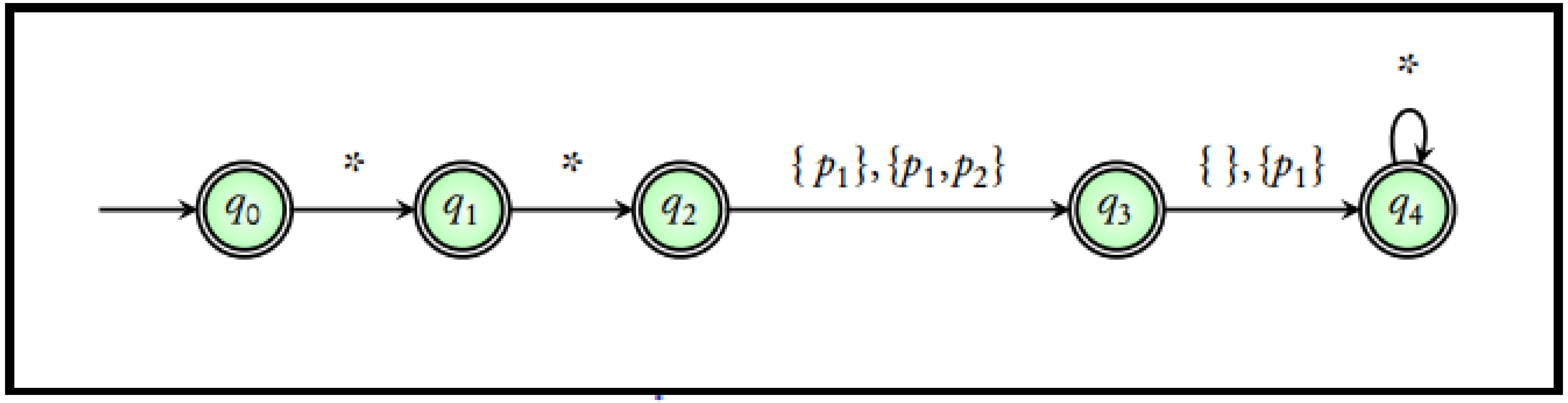


LTL To NBA

أمثلة بسيطة عن عملية التحويل بدون استخدام خوارزمية

مثال 3:

$$XX\,(p1 \wedge X \neg p2)$$

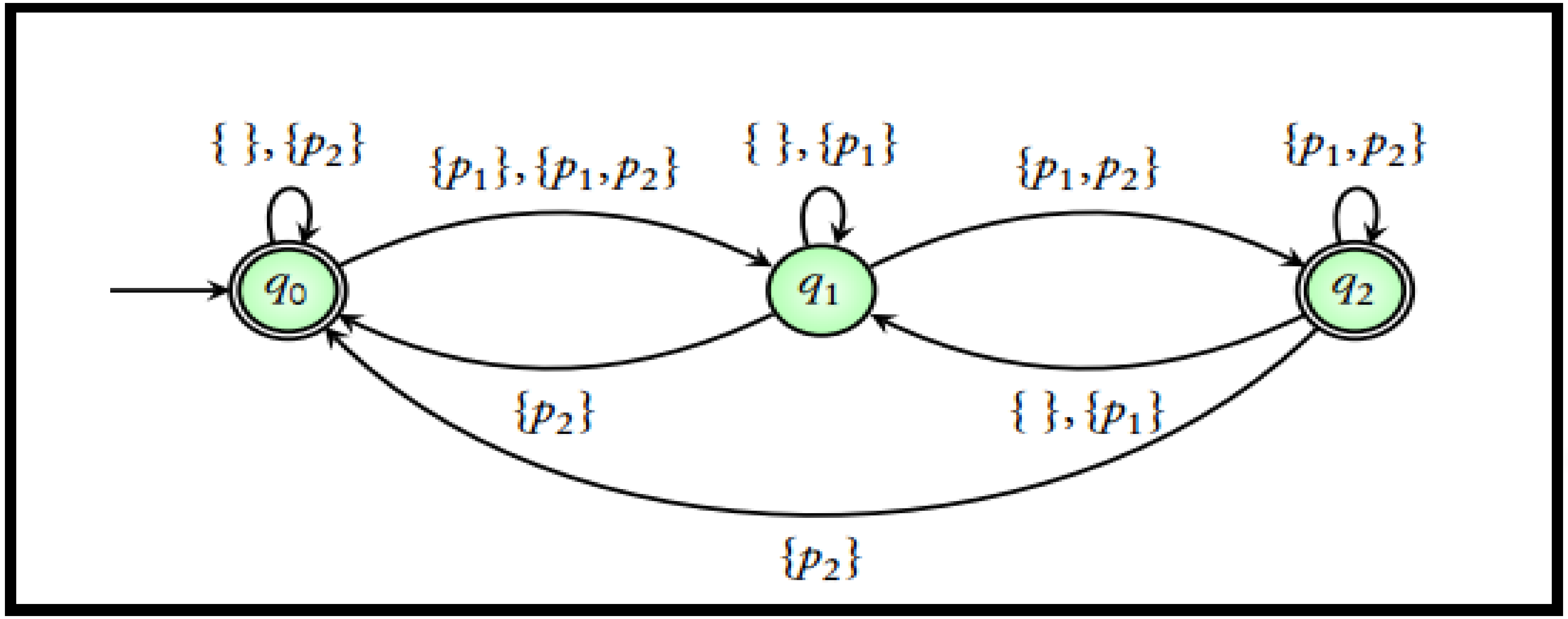


LTL To NBA

أمثلة بسيطة عن عملية التحويل بدون استخدام خوارزمية

مثال 4:

$$G\ (\ p1 \rightarrow XF\ p2)$$



LTL To NBA

خوارزمية التحويل

- Formula Expansions استخراج الـ
- تشكيل Generalized NBA (GNBA)
- التحويل من GNBA إلى NBA

LTL To NBA

Formula expansions - خوارزمية التحويل

✓ نقوم بإعادة كتابة الصيغة اعتماداً على مجموعة القواعد التالية

- $\neg\neg\varphi = \varphi$
- $\neg G\ \varphi = F\ \neg\varphi$
- $\neg F\ \varphi = G\ \neg\varphi$
- $\neg(\varphi\ U\ \psi) = (\neg\varphi)\ V\ (\neg\psi)$
- $\neg(\varphi\ V\ \psi) = (\neg\varphi)\ U\ (\neg\psi)$

- $(X\ \varphi)\ U\ (X\ \psi) \equiv X\ (\varphi\ U\ \psi)$
- $(X\ \varphi) \wedge (X\ \psi) \equiv X\ (\varphi \wedge \psi)$
- $GF\ \varphi \wedge GF\ \psi \equiv GF\ (\varphi \wedge \psi)$
- $F\ \varphi \equiv \mathbf{T}\ U\ \varphi$
- $G\ \varphi \equiv \neg F\ \neg\varphi \equiv \neg(\mathbf{T}\ U\ \neg\varphi) = \mathbf{F}\ V\ \varphi$

✓ نقوم بتقسيم الصيغة إلى مجموعة الصيغ الجزئية

✓ تحقيق عملية التوافق لكل عنصر من مجموعة الصيغ الجزئية مع كل محرف من الكلمة (Compatibility)

LTL To NBA

Compatibility - خوارزمية التحويل

Until-Compatibility

ϕ_1	*	1		0	1	
ϕ_2	1	0		0	0	
$\phi_1 \; U \; \phi_2$	1	1	1	0	0	0

LTL To NBA

Formula expansions خوارزمية التحويل - مثال لتوضيح

$\neg$ true U $\neg$ (true U p1)

الخطوة الثالثة:

	{}	{}	{p_1}	{}	{}	{p_1}	{}	{}	{p_1}
p_1	0	0	1	0	0	1	0	0	1
true	1	1	1	1	1	1	1	1	1
true U p_1	1	1	1	1	1	1	1	1	1
$\neg$ *true* U p_1	0	0	0	0	0	0	0	0	0
true U $\neg$ (*true* U p_1)	0	0	0	0	0	0	0	0	0
$\neg$ *true* U $\neg$ (*true* U p_1)	1	1	1	1	1	1	1	1	1

الخطوة الأولى: سنقوم

الخطوة الثانية: مجمو

{ {p1} ,

	{p_1}	{p_1}	{}	{}	{}	{}	{}	{}	{}
p_1	1	1	0	0	0	0	0	0	0
true	1	1	1	1	1	1	1	1	1
true U p_1	1	1	0	0	0	0	0	0	0
$\neg$ *true* U p_1	0	0	1	1	1	1	1	1	1
true U $\neg$ (*true* U p_1)	1	1	1	1	1	1	1	1	1
$\neg$ *true* U $\neg$ (*true* U p_1)	0	0	0	0	0	0	0	0	0

ue U p1)},

تصبح

LTL To NBA

Formula expansions خوارزمية التحويل - مثال لتوضح

$\neg$ true U $\neg$ (true U p1)

	$\{\}$	$\{\}$	$\{p_1\}$	$\{\}$	$\{\}$	$\{p_1\}$	$\{\}$	$\{\}$	$\{p_1\}$
p_1	0	0	1	0	0	1	0	0	1
$true$	1	1	1	1	1	1	1	1	1
$true \, \mathrm{U} \, p_1$	1	1	1	1	1	1	1	1	1
$\neg \, true \, \mathrm{U} \, p_1$	0	0	0	[illegible]	0	0	0	0	0
$true \, \mathrm{U} \neg (true \, \mathrm{U} \, p_1)$	0	0	0	0	0	0	0	0	0
$\neg \, true \, \mathrm{U} \neg (true \, \mathrm{U} \, p_1)$	1	1	1	1	1	1	1	1	1

	$\{p_1\}$	$\{p_1\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
p_1	1	1	0	0	0	0	0	0	0
$true$	1	1	1	[illegible]	1	1	1	1	1
$true \, \mathrm{U} \, p_1$	1	1	0	0	[illegible]	0	0	0	0
$\neg \, true \, \mathrm{U} \, p_1$	0	0	1	[illegible]	1	1	1	1	1
$true \, \mathrm{U} \neg (true \, \mathrm{U} \, p_1)$	1	1	1	[illegible]	1	[illegible]	1	1	1
$\neg \, true \, \mathrm{U} \neg (true \, \mathrm{U} \, p_1)$	0	0	0	0	0	0	0	0	0

LTL To NBA

خوارزمية التحويل - GNBA

- نقوم بتقسيم الصيغة إلى مجموعة الصيغ الجزئية
- حذف الحالات الغير متوافقة مع الصيغة Formula
- إضافة الحالة q0 على أنها الحالة البدائية للأوتومات
- إضافة الانتقالات بين الحالات
- نقوم بتحديد الحالات المقبولة أو الحالات النهائية (Accepting states) وفقا للعمليات الموجودة ضمن الصيغة

LTL To NBA

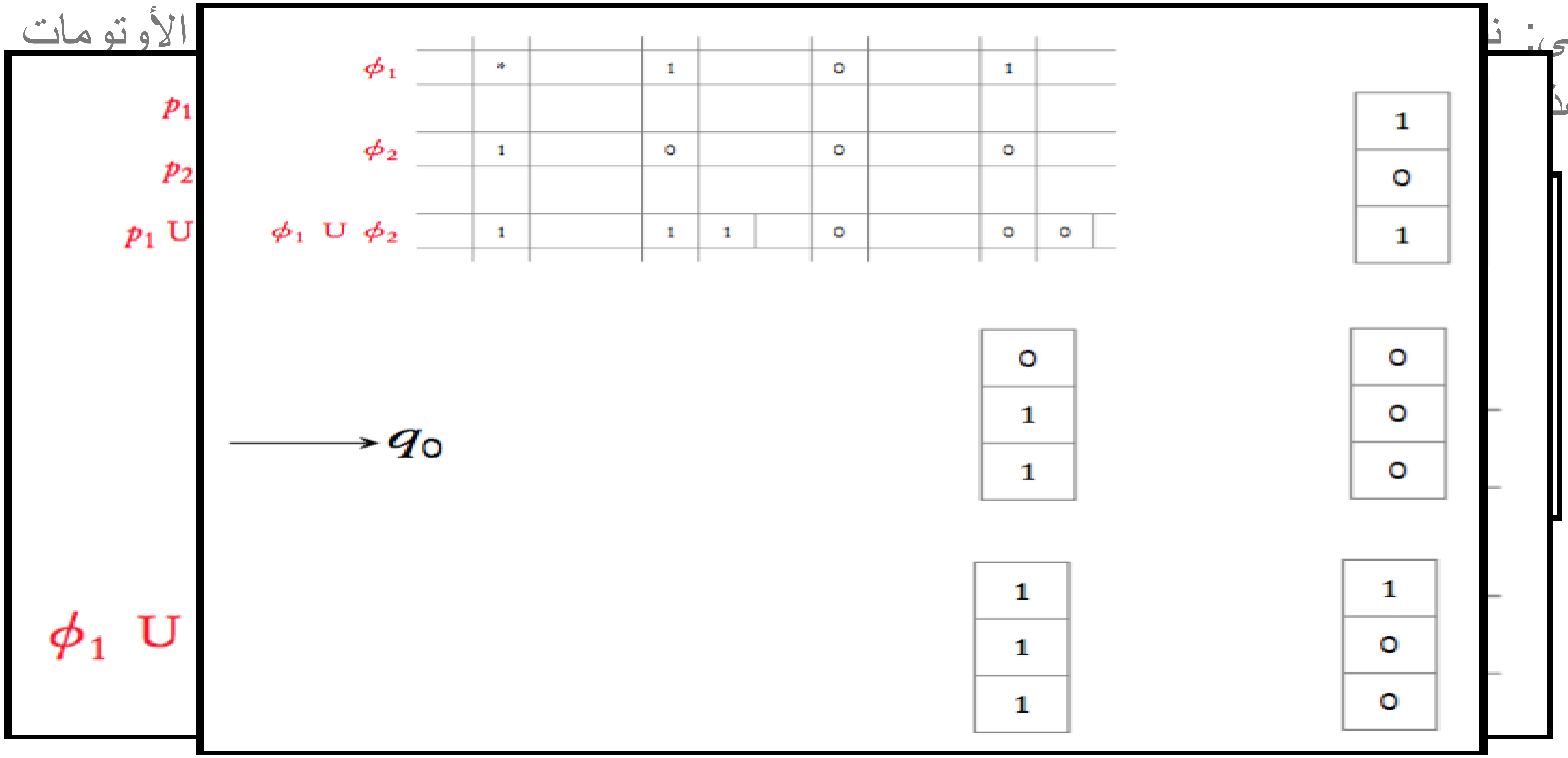
خوارزمية التحويل - مثال شامل لتوضيح عملية التحويل إلى GNBA

P1 U P2

الخطوة الثالثة: إضافة الحالات q0 الحالة البدائية Formula

الخطوة الأولى: ن … الأوتومات

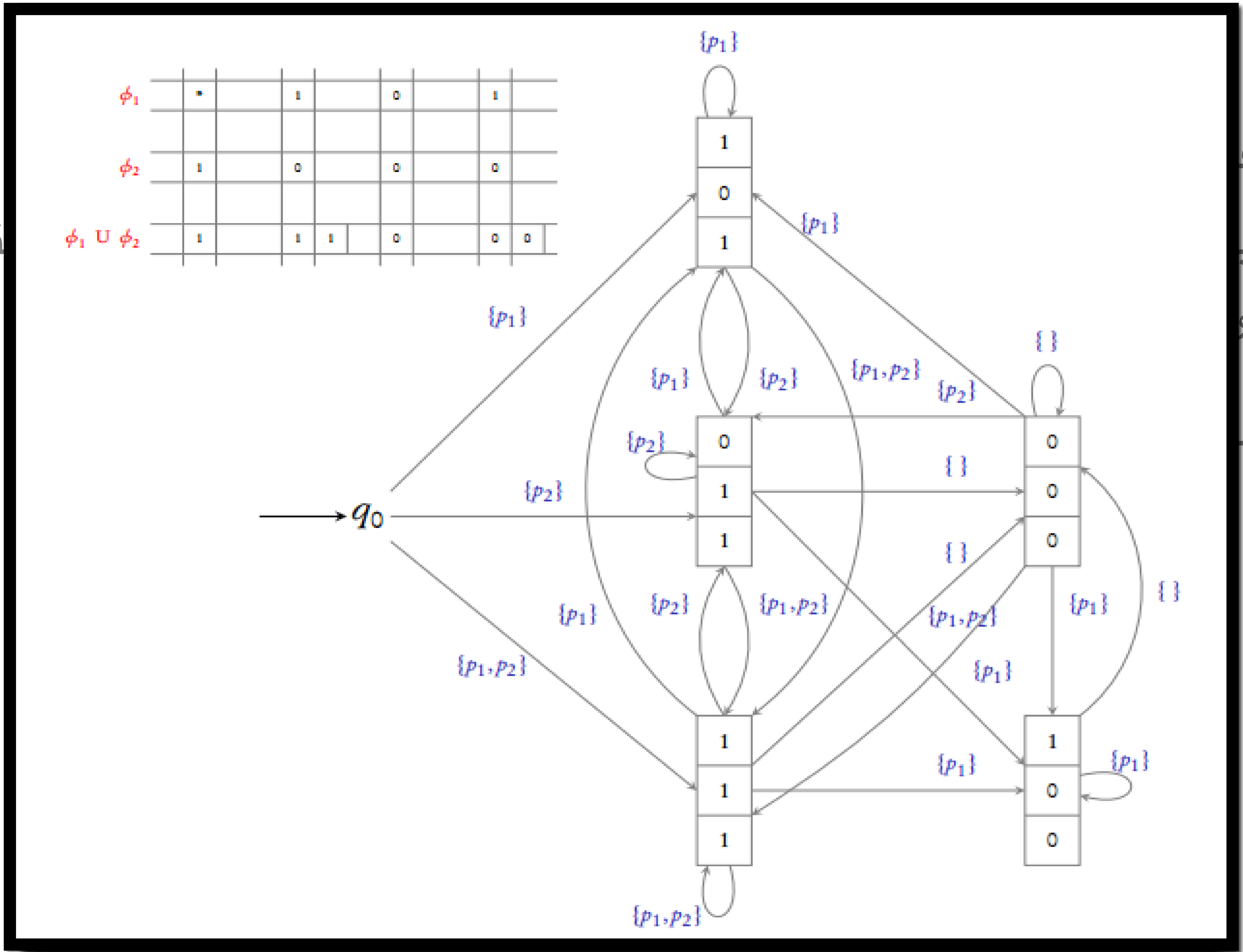
المكنة من هذ …



LTL To NBA

خوارزمية التحويل - مثال شامل لتوضيح عملية التحويل إلى GNBA

P1 U P2



الخطوة الرابعة: إضاف

✓ كل حالة لدي

تقبلها أو لا و

قق ما اذا كان الصيغة

✓ كل حرف س

✓ الانتقال من

الواحد

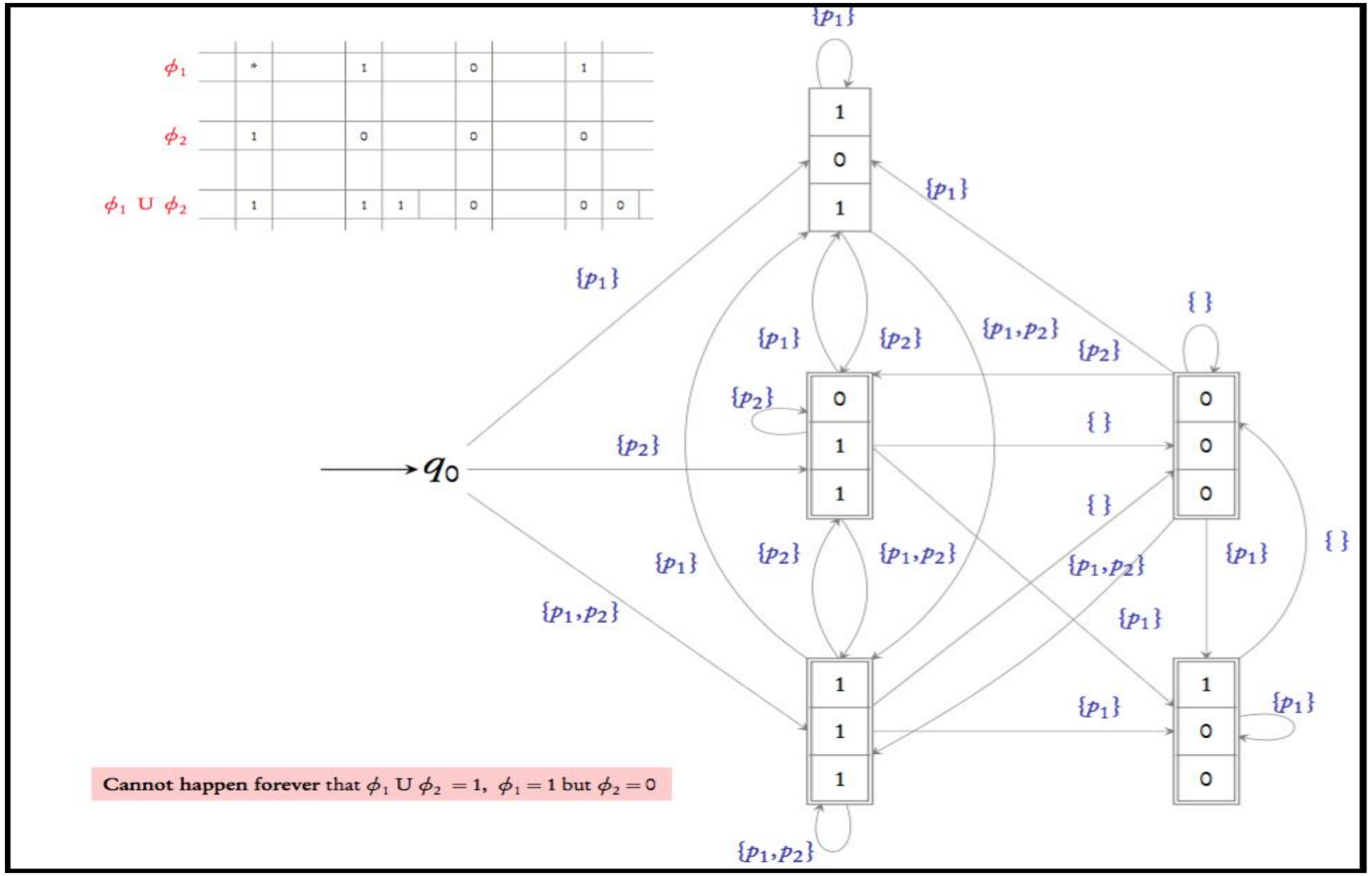
الأخيرة (P1 U P2)

LTL To NBA

خوارزمية التحويل - مثال شامل لتوضيح عملية التحويل إلى GNBA

P1 U P2

الخطوة الخامسة: تحديد الحالات النهائية في الأوتومات



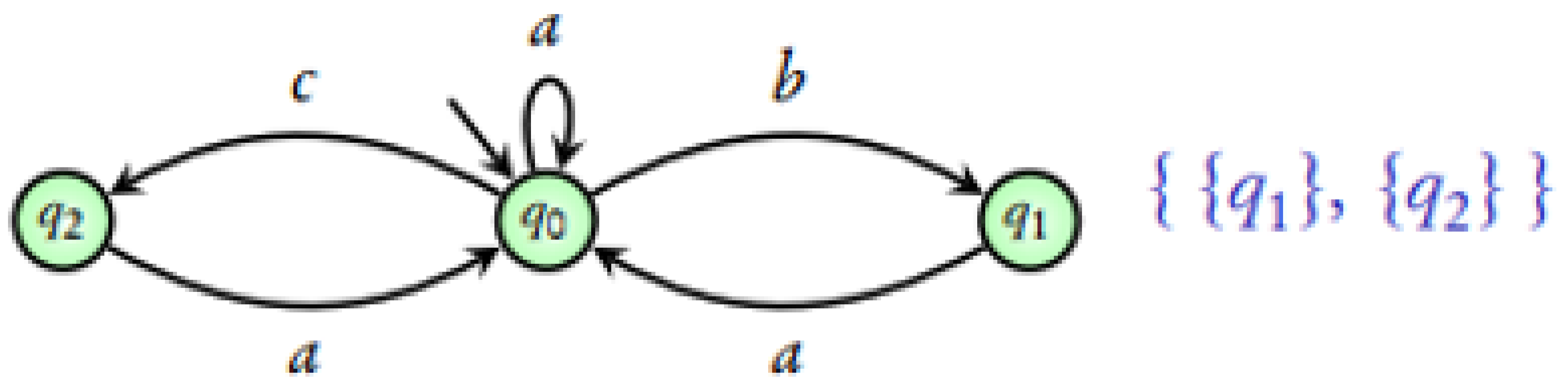
LTL To NBA

خوارزمية التحويل – التحويل من GNBA إلى NBA

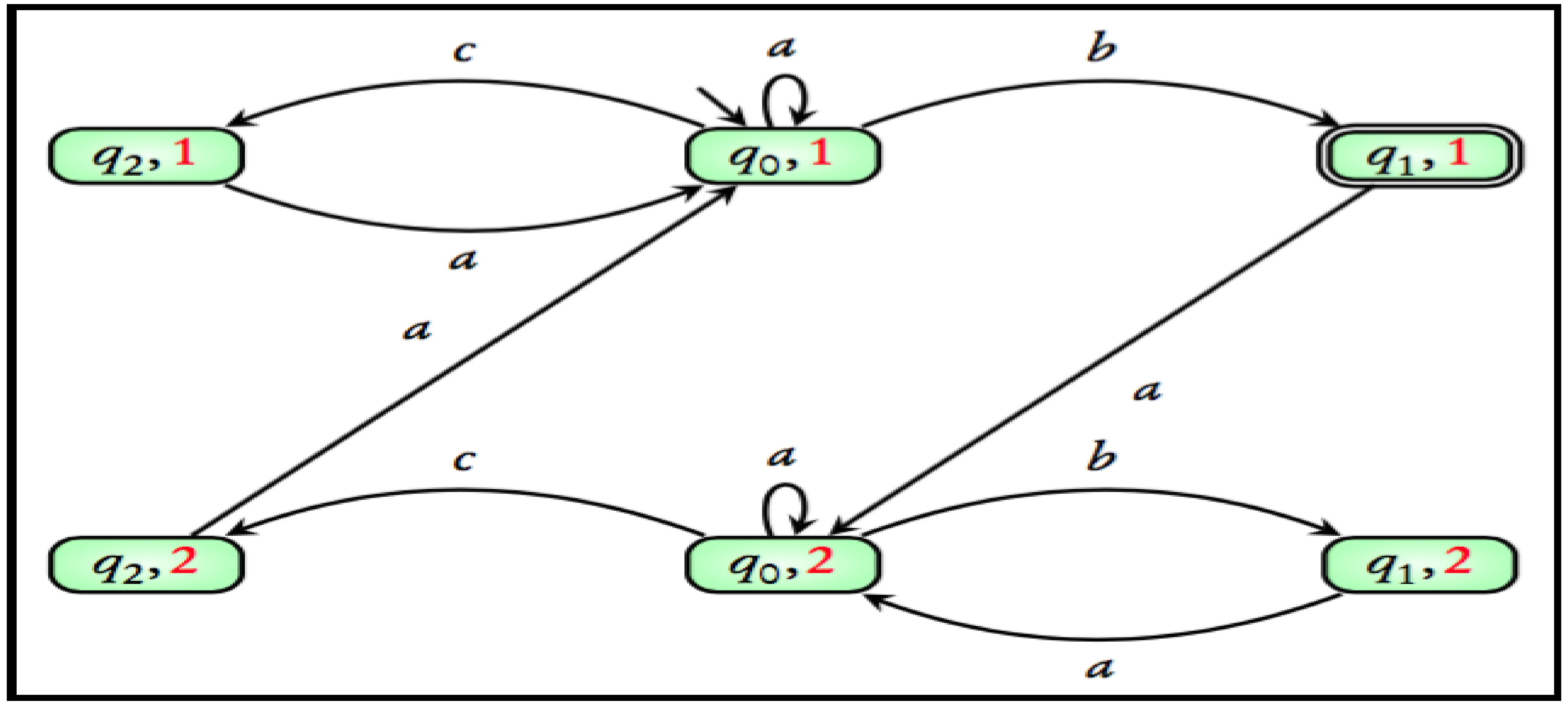
مثال:

GNBA

$(a*(b+c)a)^w$



الخطوة الثانية: الحالات المقبولة في Fi هي الحالات من المجموعة F1 بالنسخة الأولى ذات الرقم 1



كشف التضارب

خوارزميات

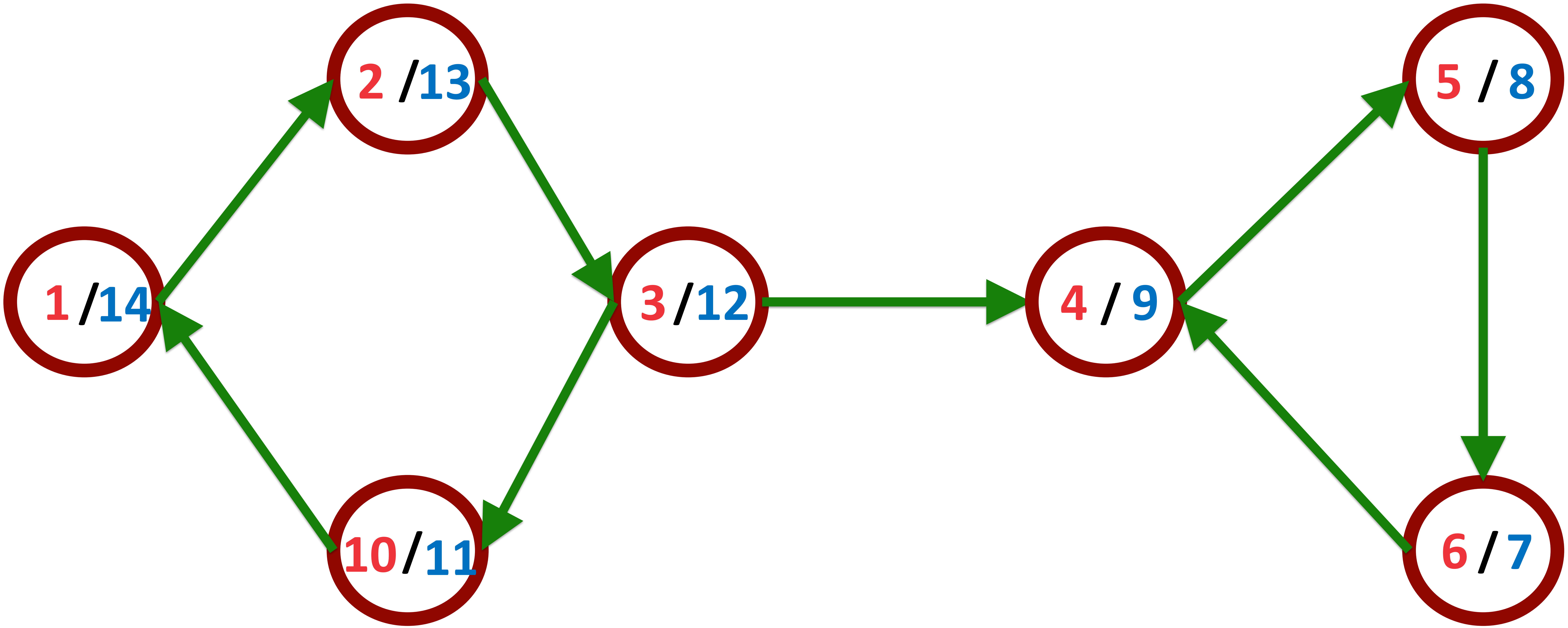
كشف التضارب

خوارزمية البحث بالعمق

```
time := 0
proc DFS(v)
    add v to Visited
    d[v] := time
    time := time + 1
    for all w ∈ succ(v) do
        if w ∉ Visited then
            DFS(w)
        end if
    end for
    f[v] := time
    time := time + 1
end proc
```

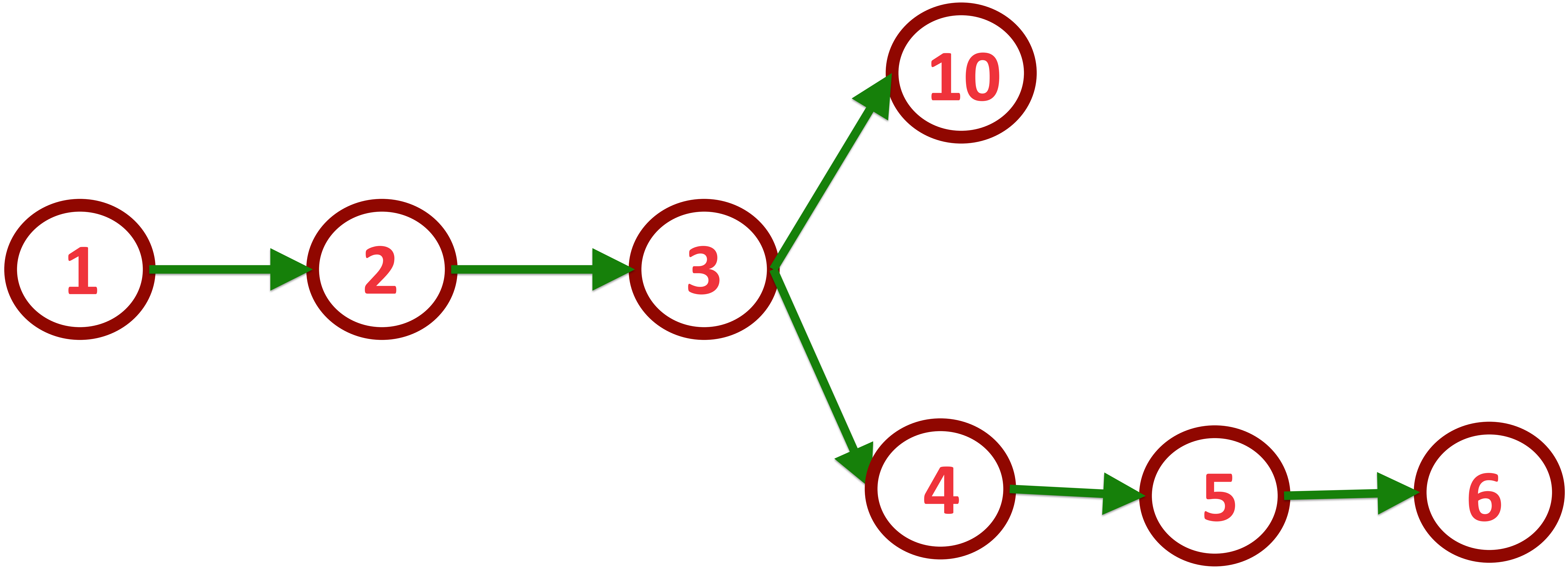
البحث بالعمق (DFS)

مثال توضيحي



شجرة البحث بالعمق (DFS Tree)

مثال توضيحي



كشف التضارب

خوارزمية Tarjan’s SCC

- ✓ **يعرّف المكوّن شديد الارتباط ضمن بيان، على أنه مجموعة العقد التي تحقق الشرط بأن يوجد طريق واحد على الأقل بين أي زوج من العقد ضمن المكوّن شديد الارتباط**
- ✓ **الفكرة الأساسية**
  - استخراج جميع المكونات شديدة الارتباط
  - يقال عن أوتومات ما بأنه غير منتهي في حال وجد حلقة واحدة على الأقل ضمنه, بحيث تحتوي هذه الحلقة على حالة نهائية واحدة على الأقل
- ✓ **حقيقة**
  - كل مكون شديد الارتباط هو شجرة فرعية ضمن شجرة البحث بالعمق الناتجة
  - لإيجاد المكوّنات شديدة الارتباط ينبغي على الخوارزمية بالتالي إيجاد جذور هذه الأشجار

Tarjan’s SCC

فكرة الخوارزمية العامة

- ✓ من أجل كل عقدة v نقوم بحساب قيمة تدعى $lowlink[v]$
- ✓ $lowlink[v]$ تمثل القيمة الدنيا بين مجموعة القيم التالية
  - ○ زمن اكتشاف العقدة v
  - ○ زمن اكتشاف العقدة w بحيث
    - • w تنتمي إلى المكون شديد الارتباط ذاته الذي تنتمي إليه العقدة v
    - • طول الطريق من العقدة v إلى العقدة w يساوي على الأقل 1
- ✓ حقيقة: يمكن القول بأن العقدة v هي جذر المكون شديد الارتباط, إذا و فقط إذا تحقق الشرط التالي
  - ○ $d[v] = lowlink[v]$

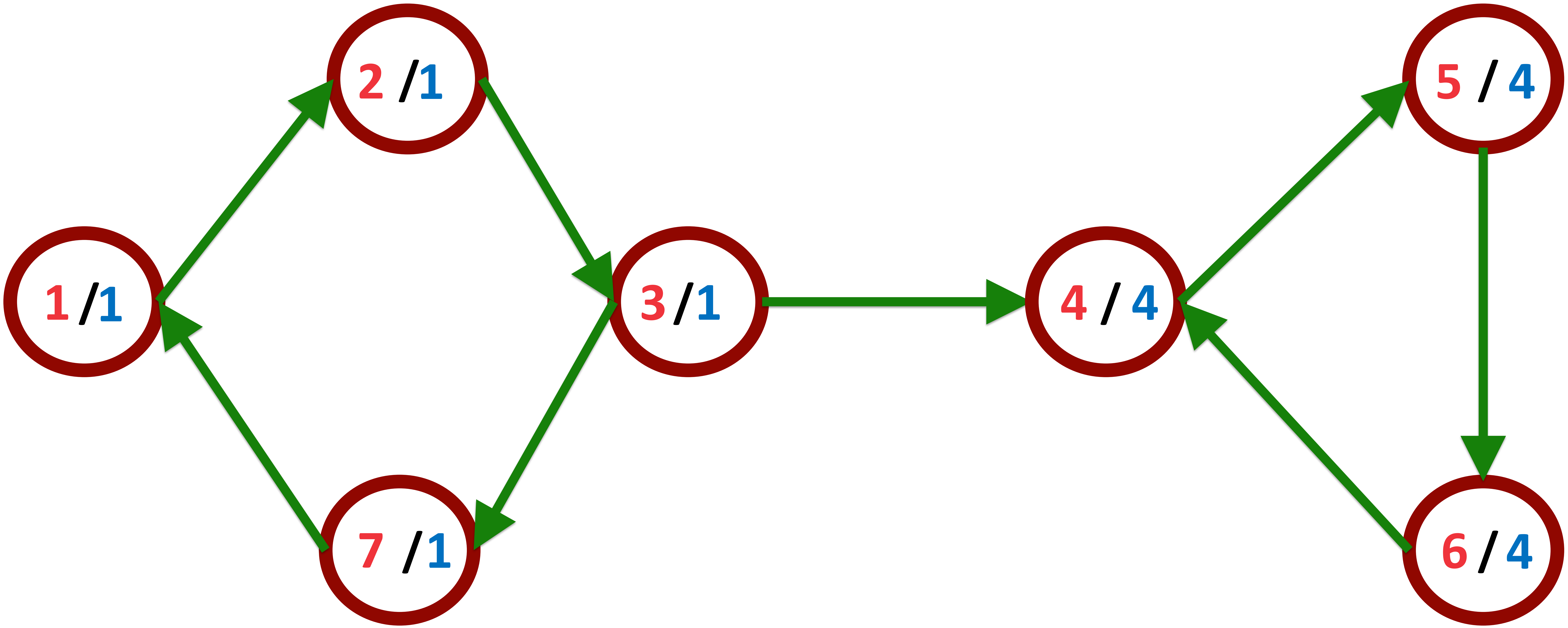
Tarjan's SCC

الخوارزمية

```
proc SCC_SEARCH(v)
    add v to Visited
    d[v] := time
    time := time + 1
    lowlink[v] := d[v]
    push v on STACK
    for all w ∈ succ(v) do
        if w ∉ Visited then
            SCC_SEARCH(w)
            lowlink[v] := min(lowlink[v], lowlink[w])
        else if d[w] < d[v] and w is on STACK then
            lowlink[v] := min(d[w], lowlink[v])
        end if
    end for
    if lowlink[v] = d[v] then
        repeat
            pop x from top of STACK
            if x ∈ F then
                terminate with “Yes”
            end if
        until x = v
    end if
end proc
```

Tarjan’s SCC

مثال توضيحي



Tarjan’s SCC

تحليل الخوارزمية

- ✓ **زمن التنفيذ:**
  - ○ خطي نسبة إلى عدد عقد البيان.
- ✓ **الذاكرة المستهلكة:**
  - ○ $\boldsymbol{Visited}$ خطية بالنسبة إلى عدد عقد البيان.
  - ○ $\boldsymbol{lowlink}$ خطية بالنسبة إلى عدد عقد البيان.
- ✓ **الأمثلة المسببة لحالات الخطأ:**
  - ○ يمكن استخراج هذه الأمثلة من المكدس المستخدم.
  - ○ يمكن حتى استخدام الخوارزمية لاستخراج عدة أمثلة مسببة لحالات الخطأ.

كشف التضارب

خوارزمية Two Sweeps

- ✓ بديل عن البحث عن المكونات شديدة الارتباط
- ✓ نقوم بإيجاد الحالات النهائية التي تقع ضمن حلقات يمكن الوصول إليها من إحدى الحالات البدائية
- ✓ الفكرة العامة تكمن في استخدام تابعي بحث بالعمق
  - البحث بالعمق الأول مهمته إيجاد جميع الحالات النهائية ضمن البيان
  - البحث بالعمق الثاني مهمته هي إيجاد الحلقة التي تقع ضمنها الحالة النهائية
- ✓ المشكلة
  - لم تعد الخوارزمية خطية بسبب الحاجة إلى زيارة العقد عدة مرات
- ✓ إصلاح مشكلة عدم الخطية
  - لتكن لدينا عقدة $\boldsymbol{v}$ إضافة إلى عقدة أخرى $\boldsymbol{u}$ فإن
    - إذا تحققت المتراجحة $\boldsymbol{f}[\boldsymbol{v}] < \boldsymbol{f}[\boldsymbol{u}]$
    - فإنه في حال كون العقدة $\boldsymbol{v}$ لا تنتمي إلى حلقة، فإنه لا توجد أي حلقات تحتوي العقدة $\boldsymbol{u}$ تحتوي على عقد يمكن الوصول إليها من العقدة $\boldsymbol{v}$

كشف التضارب

خوارزمية Two Sweeps

```
proc DFS1(v)
    add v to Visited
    for all w ∈ succ(v) do
        if w ∉ Visited then
            DFS1(w)
        end if
    end for
    if v ∈ F then
        add v to Q
    end if
end proc
```

```
proc DFS2(v, f)
    add v to Visited
    for all w ∈ succ(v) do
        if v = f then
            terminate with “Yes”
        else if w ∉ Visited then
            DFS2(w, f)
        end if
    end for
end proc
```

كشف التضارب

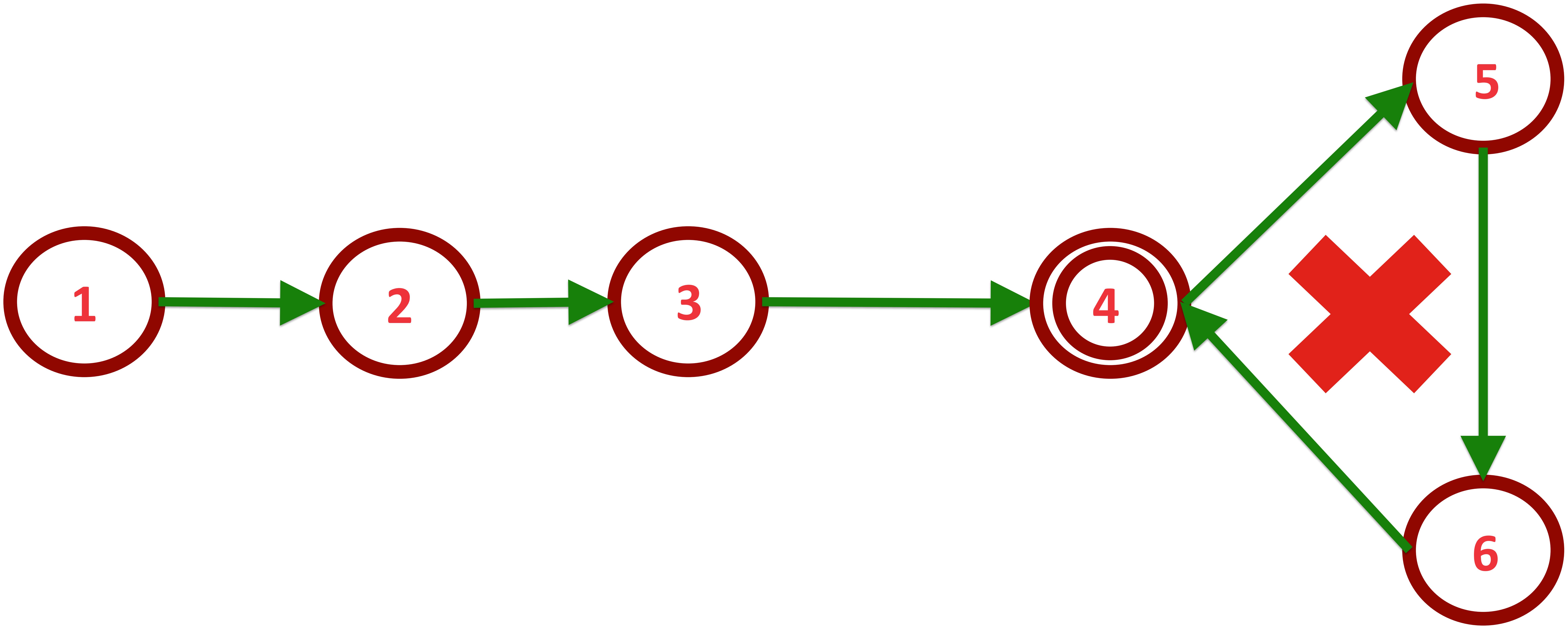
خوارزمية Two Sweeps

proc $SWEEP2(Q)$
    while $Q \neq []$ **do**
        $f := dequeue(Q)$
        $DFS2(f, f)$
    end while
    terminate with “No”
end proc

proc $DDFS(v)$
    $Q = \emptyset$
    $Visited = \emptyset$
    $DFS1(v)$
    $Visited = \emptyset$
    $SWEEP2(Q)$
end proc

Two Sweeps

مثال توضيحي



Two Sweeps

تحليل الخوارزمية

✓ **زمن التنفيذ**

- خطية (بسبب استخدام مصفوفة ***Visited*** واحدة لكل الخوارزمية مما يمنع زيارة العقدة ذاتها أكثر من مرة واحدة)

✓ **استهلاك الذاكرة**

- ***Visited*** خطية بالنسبة إلى عدد عقد البيان

✓ **الأمثلة المسببة لحالات الخطأ**

- كيف يمكن إيجاد الحالات المسببة للخطأ؟
- في الحقيقة فإننا نعلم أن شيئاً ما ضمن النظام خاطئ، و لكننا لا نعلم بالضبط ما هو، بسبب عدم قدرتنا على تحديد مثال يقوم نظامنا بمعالجته بطريقة خاطئة

كشف التضارب

خوارزمية Nested DFS

✓ الفكرة العامة

- عند الانتهاء من حالة نهائية، قم بإيقاف خوارزمية البحث بالعمق الأولى
- قم ببدأ خوارزمية البحث بالعمق الثانية، في حال إيجاد أي حلقة، فإن النظام خاطئ
- و إلا نقوم بالعودة إلى الخطوة الأولى

✓ **تشابه كثيراً الخوارزمية الأولى، مع بعض الفروقات البسيطة**

- لا تحتاج هذه الخوارزمية إلى مسح البيان كاملاً لإيجاد الخطأ
- يمكننا الوصول إلى الحالات المسببة للخطأ
  - المسار المؤدي إلى الحالة الخاطئة يكون متواجداً ضمن المكدس الخاص بخوارزمية البحث بالعمق الأولى
  - الحلقة بحد ذاتها المؤدية إلى الحالة الخاطئة تكون متواجدة ضمن المكدس الخاص بخوارزمية البحث بالعمق الثانية

كشف التضارب

خوارزمية Nested DFS

proc $DFS1(v)$
  add $(v, 0)$ to $Visited$
  for all $w \in succ(v)$ **do**
    if $(w, 0) \notin Visited$ **then**
      $DFS1(w)$
    end if
  end for
  if $v \in F$ **then**
    $DFS2(v, v)$
  end if
end proc

proc $DFS2(v, f)$
  add $(v, 1)$ to $Visited$
  for all $w \in succ(v)$ **do**
    if $v = f$ **then**
      terminate with “Yes”
    else if $(w, 1) \notin Visited$ **then**
      $DFS2(w, f)$
    end if
  end for
end proc

Nested DFS

تحليل الخوارزمية

- ✓ **تشابه الخوارزمية السابقة**
- ✓ **لسنا بحاجة إلى استخدام جدولي $\boldsymbol{Visited}$**
  - ○ نقوم بدمج كلا مصفوفتي $\boldsymbol{Visited}$ ضمن مصفوفة واحدة
    - • $(\boldsymbol{v},\mathbf{0})$ تمثل حالة العقدة $\boldsymbol{v}$ بالنسبة لخوارزمية البحث بالعمق الأولى
    - • $(\boldsymbol{v},\mathbf{1})$ تمثل حالة العقدة $\boldsymbol{v}$ بالنسبة لخوارزمية البحث بالعمق الثانية
- ✓ **التحسين على الخوارزمية السابقة**
  - ○ يمكن لهذه الخوارزمية التوقف عن سبر كامل عقد البيان بمجرد وصولها إلى مثال يقوم النظام بالتعامل معه بشكل خاطئ

المراجع المستخدمة

المراجع المستخدمة

- ✓ C. Baier and J.-P. Katoen, "Nested Depth-First Search," in *Principles of Model Checking*, London, England, Cambridge, Massachusetts, The MIT Press, 2007, pp. 203-217.
- ✓ M. Y. Vardi, "Automata-Theoretic Model Checking Revisited," 2008.
- ✓ I. Barland, J. Greiner and M. Vardi, "Using Temporal Logic to Specify Properties," 2005.
- ✓ A. Gurfinkel" ،Automata-Theoretic LTL Model-Checking ."
- ✓ K. Y. Rozier, "Linear Temporal Logic Symbolic Model Checking," *ScienceDirect,* 2010.
- ✓ V. Bloemen and J. van de Pol, "Multi-core SCC-Based LTL Model Checking," 2016.
- ✓ R. Majumdar and R. Jhala, "Software Model Checking," *ACM Computing Surveys (CSUR),* 2009.
- ✓ M. Huth and M. Ryan, LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, London: Cambridge University Press, 2004.
- ✓ B. Srivathsan, "Algorithms for LTL," *NPTEL-course,* 2015.
- ✓ B. Srivathsan, "Model-checking W-regular properties," *NPTEL-course,* 2015.