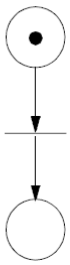
Petri Nets

- **Systems are specified as a directed bipartite graph. The two kinds of nodes in the graph:**
  1. ***Places*: they hold the distributed state of the system expressed by the presence/absence of tokens in the places.**
  2. ***Transitions*: denote the activity in the system**



- ***The state of the system*: captured by the marking of the places (number of tokens in each place)**

شبكات بتري: نظرة عامة

تُعتبر شبكات بتري أداة رياضية ورسومية قوية جداً نستخدمها في تصميم وتحليل الأنظمة المدمجة، خاصة تلك التي تحتوي على **أحداث متزامنة (concurrent events)** أو **مشاركة للموارد (shared resources)** . يمكن التفكير فيها كأنها لغة نرسم بها سلوك النظام.

مكونات شبكة بتري الأساسية

يُوصف النظام **كمخطط ثنائي موجه (directed bipartite graph)** . "ثنائي" يعني أن لدينا نوعين فقط من العقد (nodes) في هذا المخطط، و"موجه" يعني أن الأسهم لها اتجاه محدد.

هذان النوعان من العقد هما:

1. **الأماكن(Places) :**
   - تُمثل بدوائر (كما هو مبين في الرسم على اليمين).
   - يمكن التفكير في "المكان" كأنه **حالة معينة** أو **شرط** أو حتى **مورد متاح** داخل النظام.
   - "they hold the distributed state of the system expressed by the presence/absence of tokens". هذا يعني أن حالة جزء من النظام ممثلة بوجود أو عدم وجود **الرموز (Tokens)** في هذه الأماكن. الرموز هي تلك النقاط السوداء الصغيرة داخل الدوائر.
   - **مثال :**

- لنتخيل روبوتاً يقوم بتجميع قطع. يمكن أن يكون لدينا "مكان" اسمه "قطعة متوفرة". إذا كان هناك رمز (token) في هذا المكان، فهذا يعني أن هناك قطعة جاهزة للروبوت ليلتقطها.
- "مكان" آخر قد يكون "ذراع الروبوت حرة". وجود رمز فيه يعني أن الذراع ليست مشغولة بمهمة أخرى.

2. **الانتقالات(Transitions) :**
   - تُمثل عادةً بخطوط أو مستطيلات (الخط الأفقي في الرسم على اليمين).
   - "الانتقال" يمثل **حدثاً(event)** ، أو **نشاطاً(activity)** ، أو **عملية (process)** يمكن أن تحدث في النظام.
   - هذه الانتقالات هي التي تُغير حالة النظام عن طريق "استهلاك" الرموز من أماكن الإدخال و"إنتاج" رموز جديدة في أماكن الإخراج.
   - **مثال (متابعة لمثال الروبوت) :**
     - يمكن أن يكون لدينا "انتقال" اسمه "التقاط القطعة". هذا الانتقال لن "يُطلق (fire) "أو يحدث إلا إذا كان هناك رمز في "قطعة متوفرة "و رمز في "ذراع الروبوت حرة".
     - عندما يُطلق هذا الانتقال، فإنه يستهلك الرمز من "قطعة متوفرة" (لأنه تم التقاطها) ويستهلك الرمز من "ذراع الروبوت حرة" (لأنها أصبحت مشغولة)، وقد يُنتج رمزاً في مكان جديد مثل "الذراع تحمل قطعة".

حالة النظام(The State of the System)

حالة النظام الإجمالية تُلتقط بواسطة علامات الأماكن " marking of the places"

- **العلامات (Marking) :**هي ببساطة توزيع الرموز الحالي عبر جميع الأماكن في الشبكة. أي، كم عدد الرموز الموجودة في كل مكان في لحظة معينة.
- هذه العلامات هي التي تُعرفنا **بالحالة الراهنة الكاملة للنظام المدمج** .عندما يتغير توزيع الرموز (بسبب إطلاق أحد الانتقالات)، نقول إن النظام قد انتقل إلى حالة جديدة.

مثال للرسم التوضيحي على اليمين:

- نرى **مكاناً** (الدائرة العلوية) يحتوي على **رمزواحد** (النقطة السوداء)
- هذا المكان متصل بسهم إلى **انتقال** (الخط الأفقي).
- الانتقال بدوره متصل بسهم إلى **مكان آخر** (الدائرة السفلية)، وهي فارغة حالياً.
- هذا يعني: إذا كان المكان العلوي شرطاً مُسبقاً للانتقال، وكان الانتقال مُهيأً للإطلاق (قد تكون هناك شروط أخرى غير ممثلة هنا)، فعند إطلاقه سيُستهلك الرمز من المكان العلوي، وسيُنتج رمز في المكان السفلي. هذا يغير "حالة" هذا الجزء الصغير من النظام.

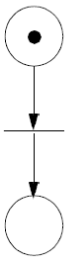
لماذا هذا مهم في تصميم الأنظمة المدمجة وهندسة الروبوت ؟

لأن الأنظمة التي نصممها غالباً ما تكون معقدة. الروبوت قد يحتاج إلى القيام بمهام متعددة في نفس الوقت (مثل التحرك، واستشعار البيئة، ومعالجة البيانات). شبكات بتري تساعدنا على:

- **نمذجة (model)** هذه العمليات المتوازية وتفاعلاتها.
- **تحليل (analyze)** النظام بحثاً عن مشاكل محتملة مثل **الجمود (deadlocks)** حيث ينتظر كل جزء من النظام جزءاً آخر إلى ما لا نهاية أو **المجاعة (starvation)** حيث لا يحصل جزء من النظام على الموارد التي يحتاجها أبداً.
- **التحقق (verify)** من أن النظام سيتصرف كما هو متوقع.

Petri Nets

- The dynamic evolution of the system: determined by the firing process of transitions.
  - A transition is enabled and may fire whenever all its predecessor places are marked.



Petri Nets

- The dynamic evolution of the system: determined by the firing process of transitions.
  - A transition is enabled and may fire whenever all its predecessor places are marked.
  - If a transition fires it removes a token from each predecessor place and adds a token to each successor place.

سوف نتعمق في **ديناميكية (dynamics)** عمل شبكات بتري، أي كيف يتطور النظام ويتغير بمرور الوقت.

تطور النظام: عملية "إطلاق" الانتقالات

"التطور الديناميكي للنظام يتحدد من خلال عملية إطلاق الانتقالات" (firing process) .

ماذا يعني هذا؟ يعني أن النظام لا يبقى ثابتاً. الحالات تتغير، والعمليات تحدث، وهذا التغيير هو نتيجة مباشرة لـ "إطلاق" أو "تفعيل" تلك الانتقالات التي تحدثنا عنها. الانتقال هو محرك التغيير في شبكة بتري.

شرط تمكين وإطلاق الانتقال(Enabling and Firing a Transition)

الشرط الأساسي لكي يكون الانتقال مستعداً للعمل:

- **"يكون الانتقال مُمكَّناً (enabled) وقد يُطلَق (fire) عندما تكون جميع أماكنه السابقة (predecessor places) معلمة (marked) "**
- **الأماكن السابقة (Predecessor Places) :**هي الأماكن التي تشير أسهم منها نحو الانتقال. يمكن التفكير فيها كأنها **المدخلات** أو **الشروط الضرورية** لحدوث النشاط الذي يمثله الانتقال.
- **معلمة (Marked)** : كما ذكرنا سابقاً، يعني أن المكان يحتوي على **رمز واحد على الأقل** (token).

.إذاً، لكي "يُطلق" انتقال ما، يجب أن تكون كل الشروط الممثلة بأماكن الإدخال الخاصة به متحققة (أي، تحتوي على رموز)

مثال من عالم الروبوتات :لنتخيل انتقالاً اسمه "تحريك الذراع إلى الموقع X " .

- قد يكون له مكان سابق (شرط) اسمه "الذراع حرة".
- ومكان سابق آخر اسمه "تم حساب المسار إلى X " .هذا الانتقال "تحريك الذراع إلى الموقع X " لن يكون مُمكَّناً (ولن يستطيع الإطلاق) إلا إذا كان هناك رمز في "الذراع حرة "و رمز في "تم حساب المسار إلى X". إذا كان أحد الشرطين غير متحقق (أحد المكانين لا يحتوي على رمز)، يبقى الانتقال خاملاً.

الرسم التوضيحي في الشريحة الأولى يظهر هذا: المكان العلوي (السابق) يحتوي على رمز، مما يجعل الانتقال (الخط الأفقي) مُمكَّناً وجاهزاً .للإطلاق

ماذا يحدث عندما "يُطلَق" الانتقال(The Firing Action) ؟

- **"إذا أُطلِق انتقال ما، فإنه يزيل رمزاً من كل مكان سابق (predecessor place) ويضيف رمزاً إلى كل مكان لاحق (successor place) ".**

هنا تكمن ديناميكية التغيير:

- **إزالة الرموز من الأماكن السابقة :**هذا يمثل "استهلاك" الشروط أو الموارد التي كانت ضرورية لحدوث النشاط.

- **إضافة الرموز إلى الأماكن اللاحقة (Successor Places)**: الأماكن اللاحقة هي تلك التي تشير أسهم من الانتقال إليها .تمثل هذه **النتائج** أو **الحالات الجديدة** التي تنشأ بعد حدوث النشاط.

نعود لمثال الروبوت "تحريك الذراع إلى الموقع X": عندما يُطلق هذا الانتقال (لأن الشروط تحققت):

1. **يُزال** الرمز من "الذراع حرة" (لأن الذراع الآن بدأت بالحركة وأصبحت مشغولة).
2. **يُزال** الرمز من "تم حساب المسار إلى X" (لأنه تم استخدام هذا المسار).
3. **يُضاف** رمز إلى مكان لاحق (مخرج) قد يكون اسمه "الذراع تتحرك".
4. وربما **يُضاف** رمز إلى مكان لاحق آخر مثل "الذراع في الموقعX "بعد اكتمال الحركة، لكن هذا قد يكون انتقالاً آخر أكثر تفصيلاً.

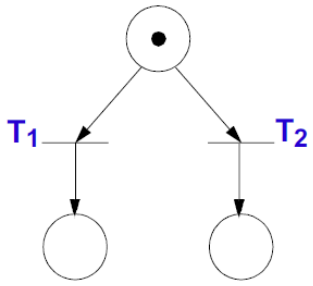
- في البداية كان الرمز في المكان العلوي (السابق).
- بعد أن أُطلق الانتقال، تم إزالة الرمز من المكان العلوي (أصبح فارغاً).
- وتم إضافة رمز إلى المكان السفلي (اللاحق).

هذه هي **دورة الحياة** في شبكة بتري: الأماكن تحتفظ بالحالات (الرموز)، والانتقالات تُمكن بناءً على هذه الحالات، وعندما تُطلق الانتقالات، تُحدث تغييراً في توزيع الرموز، مما يؤدي إلى حالة جديدة للنظام، وهكذا دواليك. هذا التتابع من إطلاق الانتقالات وتغير العلامات . هو ما يمثل سلوك النظام المدمج الذي نصممه.

نلاحظ كيف تصف هذه القواعد البسيطة سلوكاً يمكن أن يكون معقداً للغاية و هذا هو جمال وقوة شبكات بتري.

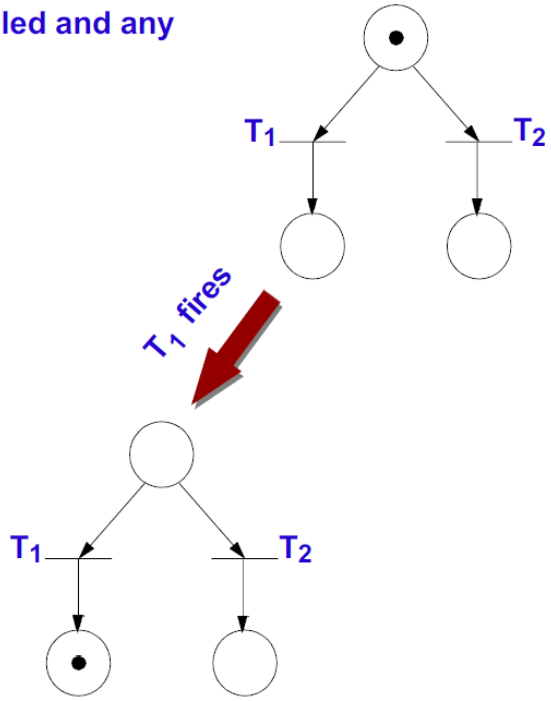
Nondeterminism

- **Both T1 and T2 are enabled and any of the two may fire.**



Nondeterminism

- **Both T1 and T2 are enabled and any of the two may fire.**



سنتناول الآن مفهوم واقعي في عالم الأنظمة المدمجة، ألا وهو **اللاحتمية (Nondeterminism)** . هذا المفهوم يضيف طبقة أخرى من الواقعية على نماذجنا.

اللاحتمية: عندما يكون للنظام خيارات

النقطة الأساسية هنا هي" :**كلا من T1 و T2 مُمكَّنان وأي منهما يمكن أن يُطلَق**".

ماذا يعني هذا في سياق شبكات بتري؟ لنتذكر القاعدة التي تعلمناها: "يكون الانتقال مُمكَّناً وقد يُطلَق عندما تكون جميع أماكنه السابقة معلمة".

بالنظر إلى الرسم في الشريحة:

- لدينا مكان علوي (سوف نسميه **P0** لسهولة الشرح) يحتوي على **رمز واحد**.
- هذا المكان P0 هو **مكان سابق (predecessor)** لكل من الانتقال **T1**والانتقال **T2** أي أن سهماً يخرج من P0 إلىT1 ، وسهماً آخر يخرج من P0 إلى T2
- بما أن P0 يحتوي على رمز، فإنه يلبي شرط التمكين لكل من T1 و T2 في نفس الوقت.

هنا تظهر **اللاحتمية** :النظام الآن في حالة يمكن فيها أن يحدث أحد أمرين (أو أكثر، في شبكات أكبر): إما أن يُطلقT1 ، أو أن يُطلق T2 . النموذج الأساسي لشبكة بتري لا يُملي علينا أيهما سيُطلق أولاً أو بالتأكيد هناك **خيار**.

مثال لتوضيح الفكرة :لنتخيل أن الروبوت الخاص بنا (النظام) وصل إلى نقطة قرار(P0 لديه رمز = "الروبوت عند نقطة القرار").

- T1يمثل "اتجه يساراً".
- T2يمثل "اتجه يميناً ". كلا الخيارين متاحان الآن. اللاحتمية تعني أن الروبوت قد يتجه يساراً، أو قد يتجه يميناً. الشبكة نفسها لا تفضل أحدهما على الآخر ما لم نضف قواعد إضافية (مثل الأولويات، وهو موضوع متقدم).

أحد المسارات الممكنة: إطلاقT1

الشريحة الثانية توضح ما يحدث إذا تم اختيار أحد المسارات، وفي هذه الحالة، إذا **أُطلِقT1 :**

1. **الحالة الابتدائية :**كما رأينا، P0 لديه رمز، و T1 و T2 كلاهما مُمكَّن.
2. **الحدث :**السهم الأحمر الكبير يشير إلى أن "T1 يُطلَق (fires) " .هذا يعني أنه من بين الخيارات المتاحة، تم تنفيذ النشاط الذي يمثله T1 .
3. **الحالة الناتجة :**
   - يتم **إزالة الرمز** من المكان السابق المشترك P0 (لأن T1 استهلك الشرط/المورد).
   - يتم **إضافة رمز** إلى المكان اللاحق لT1 (الدائرة السفلية اليسرى، لنسميهاP1).
   - **ملاحظة مهمة جداً :**بما أن الرمز قد أُزيل منP0 ، فإن الانتقال T2 **لم يعد مُمكَّناً** الآن حيث لم يعد شرطه المسبق (وجود رمز في P0) متحققاً. هذا ما يُعرف في شبكات بتري بـ **التعارض أو التنازع(Conflict)** ؛ T1 و T2 كانا في تعارض على المورد (الرمز) فيP0 ، وفاز T1 به (في هذا السيناريو).

لو كان T2 هو الذي أُطلق بدلاً منT1 ، لكان الرمز قد انتقل إلى المكان اللاحق لـ T2 (الدائرة السفلية اليمنى، P2) ، وكان T1 هو الذي سيصبح غير مُمكَّن.

أهمية اللاحتمية في تصميم الأنظمة المدمجة وهندسة الروبوت:

الأنظمة التي نتعامل معها نادراً ما تكون خطية تماماً. اللاحتمية ضرورية لنمذجة جوانب مثل:

- **الخيارات الحقيقية في النظام :**مثل استجابة الروبوت لمدخلات المستخدم أو لبيانات استشعار غير متوقعة. قد يكون هناك عدة استجابات صحيحة.
- **التنافس على الموارد المشتركة :**لنفترض لدينا معالج صغري (مورد مشترك) تحتاجه عدة عمليات (انتقالات) في نفس الوقت. واحدة فقط يمكنها استخدامه في لحظة معينة. شبكات بتري يمكنها نمذجة هذا التنافس.
- **تحليل كافة السلوكيات الممكنة :**عندما نصمم نظاماً، ونموذجه بشبكة بتري يحتوي على لاحتمية، يجب علينا تحليل جميع المسارات الممكنة التي يمكن أن يسلكها النظام. هل تؤدي جميعها إلى حالات آمنة ومرغوبة؟ هل يمكن أن يؤدي تسلسل معين من الخيارات إلى جمود(deadlock) ؟

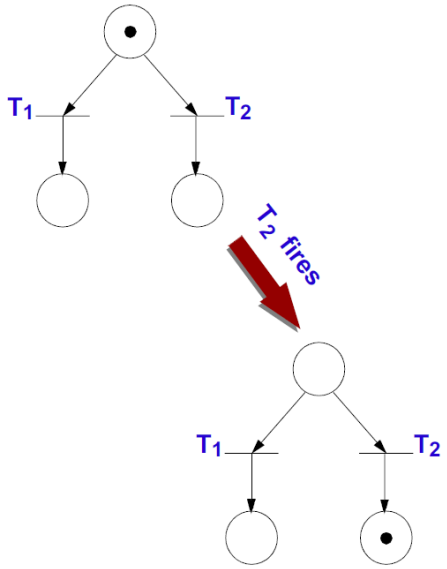
- **المرونة في التفاعل**:أحياناً، لا يهم الترتيب الدقيق لبعض العمليات غير المعتمدة على بعضها البعض.

فهم اللاحتمية يساعدنا على تصميم أنظمة أكثر قوة وقدرة على التعامل مع سيناريوهات متنوعة. إنه يدفعنا للتفكير في "**ماذا لو**؟" عند كل نقطة قرار في النظام.

اللاحتمية هي التي تجعل النماذج أقرب إلى سلوك الأنظمة المعقدة في العالم الحقيقي

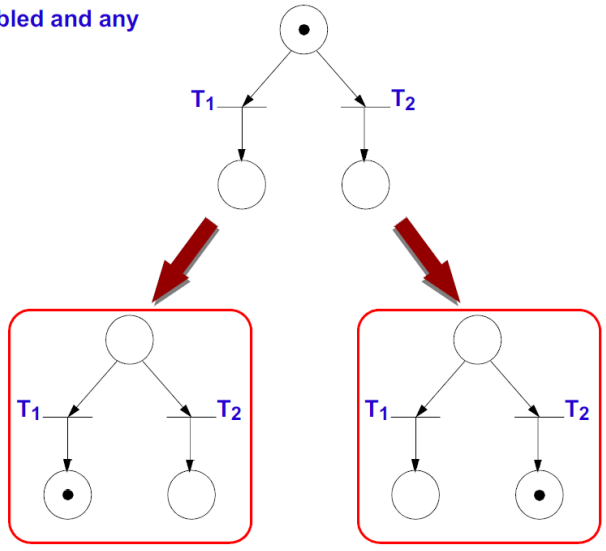
Nondeterminism

- **Both T1 and T2 are enabled and any of the two may fire.**



Nondeterminism

- **Both T1 and T2 are enabled and any of the two may fire.**
- **Any of these two states might be the next state.**



المسار البديل : إطلاقT2

- **"كلا من T1 و T2 مُمكَّنان وأي منهما يمكن أن يُطلَق "**.هذه هي الحالة الابتدائية نفسها: المكان العلوي P0 يحتوي على رمز، مما يُمكن كلاً من T1 وT2 .
- لكن هذه المرة، **T2"يُطلَق(fires)" .**
- **النتيجة؟**
  - تماماً كما في حالة إطلاقT1 ، المكان P0 (المكان السابق المشترك) يفقد الرمز الخاص به لأنه تم استهلاكه بواسطةT2 .
  - المكان اللاحق لـ T2 (الدائرة السفلية اليمنى، P2) هو الذي يحصل على الرمز.
  - والآن، بما أن P0 أصبح فارغاً، فإن T1 (الذي كان مُمكَّناً أيضاً) لم يعد كذلك. لقد حُسم **التعارض (Conflict)** لصالح T2 هذه المرة.

هذا يوضح أنه من نفس نقطة البداية، كان يمكن للنظام أن يسلك مساراً مختلفاً تماماً.

ملخص الخيارات - حالتان تاليتان محتملتان

"أي من هاتين الحالتين يمكن أن تكون الحالة التالية" (Any of these two states might be the next state) .

هذا هو جوهر **اللاحتمية** في هذا السياق: من حالة نظام واحدة، يمكن أن يتطور النظام إلى واحدة من عدة حالات تالية محتملة. لا يوجد شيء في نموذج شبكة بتري الأساسي (حتى الآن) يفرض اختيار T1 على T2 أو العكس.

ماذا يعني هذا لنا كمهندسين نصمم أنظمة الروبوت والأنظمة المدمجة؟

يعني هذا أنه عندما يحتوي نموذجنا على مثل هذه النقاط اللاحتمية (أو نقاط التعارض)، يجب أن نأخذ في الاعتبار **جميع** المسارات الممكنة التي يمكن أن يتخذها النظام:

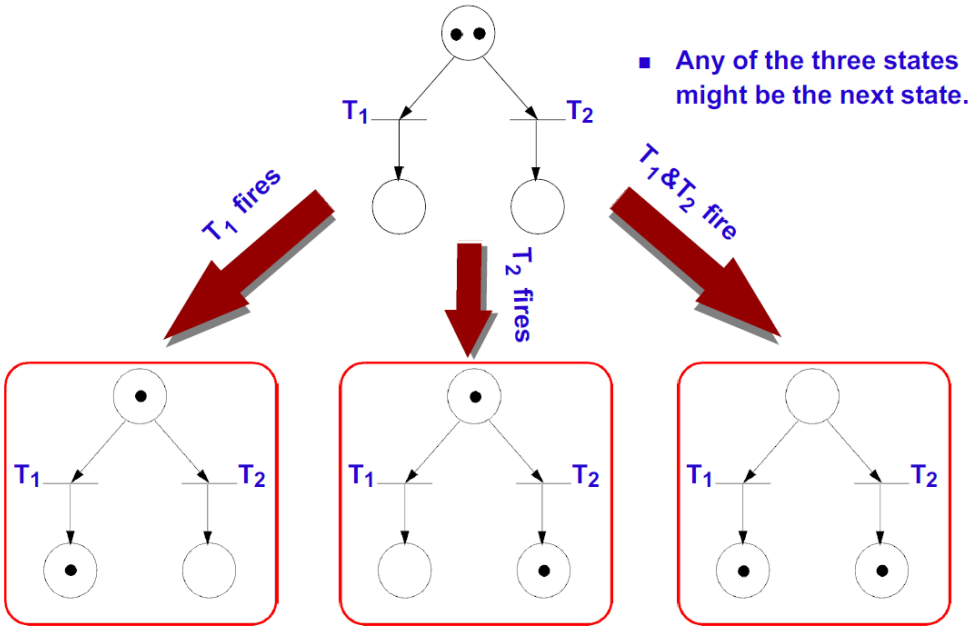
- هل تؤدي جميع هذه المسارات إلى سلوك مقبول للنظام؟
- هل يمكن لأحد الخيارات أن يقود النظام إلى حالة غير مرغوب فيها (مثل جمود أو حالة خطأ)؟
- هل نحتاج إلى إضافة آليات للتحكم في هذه الخيارات، مثل إعطاء أولويات لبعض الانتقالات على الأخرى في مواقف معينة؟

على سبيل المثال، إذا كان P0 يمثل "وجود طلب خدمة" و T1 يمثل "معالجة الطلب بواسطة الخادم الرئيسي" و T2 يمثل "معالجة الطلب بواسطة الخادم الاحتياطي"، فإن اللاحتمية هنا قد تكون مرغوبة (إذا كان أي خادم يمكنه التعامل مع الطلب). لكن قد نحتاج إلى تحليل ما إذا كان أحد الخادمين أفضل من الآخر في ظروف معينة.

فهم هذه الخيارات المتعددة أمر بالغ الأهمية لتصميم أنظمة قوية وموثوقة وقادرة على التعامل مع مختلف السيناريوهات التي قد تواجهها في العالم الحقيقي.

تمثل فكرة اللاحتمية قدرة شبكات بتري على نمذجة السلوكيات المعقدة والخيارات المتأصلة في العديد من الأنظمة.

Nondeterminism



توضح هذه الشريحة فرقاً دقيقاً ولكن حاسماً بين مفهومين :**التعارض (Conflict)** الذي رأيناه للتو، ومفهوم جديد هو **التوازي (Concurrency)**.

من التعارض إلى التوازي:

أولاً، لنلاحظ الفرق الجوهري في الحالة الابتدائية:

- في الشرائح السابقة، كان المكان العلوي (P0) يحتوي على **رمز واحد**.
- في هذه الشريحة، المكان العلوي P0 يحتوي على **رمزين (two tokens)** هذا التغيير البسيط يقلب الموازين تماماً

ماذا يعني وجود رمزين؟ يعني أن لدينا "موردين" أو "شرطين" متوفرين.

لنرى الآن كيف يؤثر هذا على الانتقالات T1 وT2 :

النتيجة الأولى (اليسار) "T1 يُطلَق":

- T1يحتاج إلى رمز واحد في P0 لكي يُطلَق. بما أن هناك رمزين، فالشرط متحقق.
- يُطلق T1 ، فيستهلك **رمزاً واحداً** من P0 ويضيف رمزاً واحداً إلى مكانه اللاحق P1
- **الحالة الناتجة :**يتبقى **رمز واحد** في P0 ، ويصبح هناك رمز واحد في P1
- **نقطة مهمة جداً :**بما أنه لا يزال هناك رمز في P0 ، فإن الانتقال T2 **لا يزال مُمكَّناً**. إطلاق T1 لم يمنع T2

2. **النتيجة الثانية (الوسط) "T2 يُطلَق":**

- بنفس المنطق، T2يستهلك **رمزاً واحداً** من P0 ويضيف رمزاً إلى مكانه اللاحق P2
- **الحالة الناتجة :يتبقى رمزواحد** فيP0 ، ويصبح هناك رمز واحد في P2
- ومرة أخرى، بما أنه لا يزال هناك رمز فيP0 ، فإن T1 **لا يزال مُمكَّناً**

حتى الآن، يبدو الأمر وكأنه لاحتمية، حيث يمكننا اختيار إطلاق T1 أو T2 .لكن الفرق الجوهري هو أن الانتقالين **لم يعودا في تعارض** .في السابق، كانا يتنافسان على مورد واحد. الآن، هناك ما يكفي من الموارد لكليهما. وهذا ما يقودنا إلى النتيجة الثالثة.

3. **النتيجة الثالثة (اليمين) "T1 و T2 يُطلقان"** :هذه هي النقطة التي توضح مفهوم **التوازي (Concurrency)** .
   - بما أن T1 و T2 ليسا في تعارض (أي أن إطلاق أحدهما لا يمنع إطلاق الآخر)، فمن الممكن أن يُطلَق كلاهما.
   - في نموذج شبكات بتري الكلاسيكي، تتم عمليات الإطلاق بشكل متسلسل (واحد تلو الآخر). لذا، للوصول إلى هذه الحالة، يمكن أن يحدث أحد أمرين :
     - T1يُطلق أولاً، ثم يُطلقT2
     - T2يُطلق أولاً، ثم يُطلقT1
   - في كلتا الحالتين، تكون النتيجة النهائية واحدة: يتم استهلاك **كلا الرمزين** منP0 ، ويتم وضع رمز في P1 ورمز فيP2
   - T1و T2 هما انتقالان **مستقلان** ويمكن تنفيذهما **بالتوازي** .في نظام حقيقي متعدد المعالجات، يمكن أن يحدثا في نفس اللحظة الزمنية. في نموذجنا، يعني ذلك أن ترتيب حدوثهما غير مهم للوصول إلى النتيجة النهائية.

خلاصة الفرق: التعارض مقابل التوازي

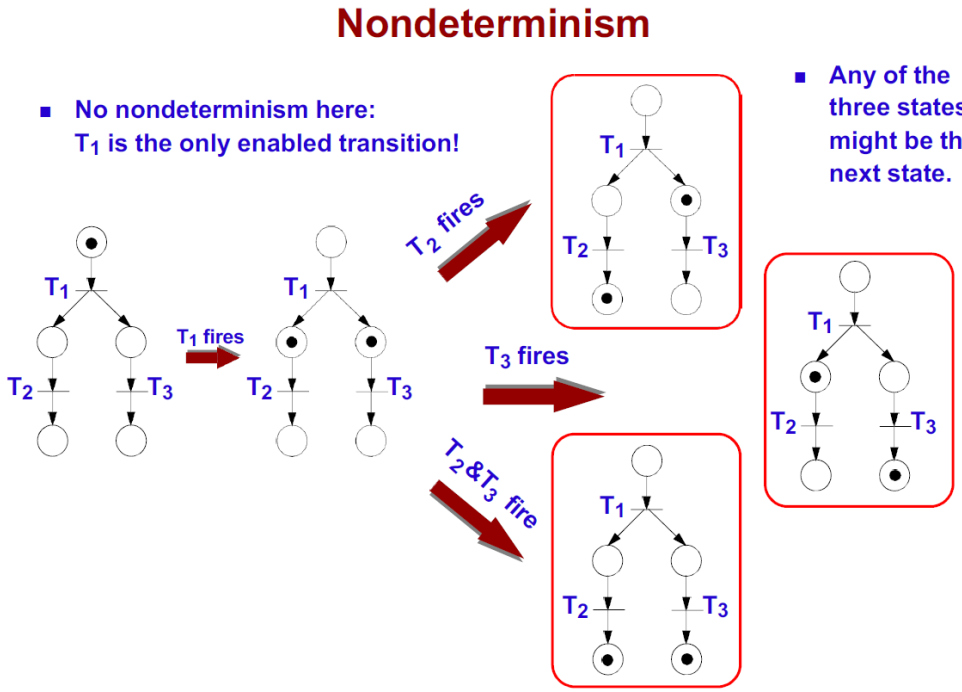
- **التعارض (Conflict)** :(رمز واحد)
  - T1و T2 كلاهما مُمكَّن.
  - إطلاق أحدهما **يمنع** إطلاق الآخر.
  - يمثل **اختياراً حصرياً (exclusive choice)** : إما هذا أو ذاك.
- **التوازي (Concurrency)** (رمزان)
  - T1و T2 كلاهما مُمكَّن.
  - إطلاق أحدهما **لا يمنع** إطلاق الآخر.
  - يمثل **إجراءات مستقلة (independent actions)** يمكن أن تحدث بأي ترتيب أو في نفس الوقت.

مثال من هندسة الروبوت :لنتخيل أن الروبوت لديه بطاريتان (رمزان في" P0 =بطارية مشحونة متاحة").

- T1 = "تشغيل نظام الملاحة"

- T2="تشغيل نظام الالتقاط " .
- كل نظام يحتاج بطارية واحدة.
- يمكن للروبوت أن يشغل نظام الملاحة أولاً (الحالة اليسرى)، وسيظل لديه بطارية لتشغيل نظام الالتقاط.
- أو يمكنه تشغيل نظام الالتقاط أولاً (الحالة الوسطى)، وسيظل لديه بطارية لتشغيل نظام الملاحة.
- في النهاية، يمكن تشغيل كلا النظامين (الحالة اليمنى)، وهذا يمثل حالة التشغيل الكامل بالتوازي.

هذه الشريحة تعلمنا كيفية تمييز ما إذا كانت الانتقالات المُمكَّنة في تعارض حقيقي أم أنها ببساطة أنشطة متوازية يمكن أن تحدث بشكل مستقل. هذا التمييز أساسي لتصميم وتحليل الأنظمة المعقدة متعددة المهام.



هذه الشريحة تجمع كل ما تعلمناه في سيناريو واحد متكامل يوضح كيف يمكن لسلوك النظام أن يتغير من **حتمي (deterministic)** إلى **لاحتمي ومتوازي (nondeterministic and concurrent)**

سوف نحلل سلوك النظام خطوة بخطوة:

المرحلة الأولى: البداية الحتمية(Deterministic Start)

- **الحالة الابتدائية** :بالنظر إلى أقصى اليسار. لدينا رمز واحد في المكان العلوي(P0) ، وهذا المكان هو المكان السابق **فقط** للانتقال T1. لا يوجد أي انتقال آخر مُمكَّن.
- **"No nondeterminism here: T1 is the only enabled transition!"**

- في هذه اللحظة، ليس لدى النظام أي خيار. المسار الوحيد الممكن للمضي قدماً هو إطلاق T1 .سلوك النظام في هذه الخطوة **حتمي 100%.**

- **بعد إطلاق T1** :
  - يستهلك T1 الرمز من P0
  - T1 له سهمان خارجان، واحد باتجاه مكان (سوف نسميه P2)هو السابق لـT2 ، والآخر باتجاه مكان (P3) هو السابق لـ T3
  - لذلك، عند إطلاقه، يضع T1 رمزاً في P2 و رمزاً فيP3
  - T1هنا يعمل كـ "**مُوزّع** "، حيث يبدأ مسارين أو عمليتين بشكل متزامن.

المرحلة الثانية: ظهور التوازي واللاحتمية

- **الحالة الوسطى (بعد إطلاق T1)**: لدينا الآن رمز في P2 ورمز فيP3
- **ما هي الانتقالات المُمكّنة الآن؟**
  - T2مُمكّن لأن مكانه السابق P2 يحتوي على رمز.
  - T3مُمكّن لأن مكانه السابق P3 يحتوي على رمز.
- **وهنا السؤال الجوهري** :هل T2 و T3 في **تعارض** أم **توازٍ**؟
  - الجواب: إنهما في **توازٍ (concurrent)** ،لأنهما يعتمدان على موردين (رمزين) مختلفين في مكانين مختلفين(P2 وP3) إطلاق T2 يستهلك الرمز من P2 ولكنه لا يؤثر على الرمز فيP3 ، والعكس صحيح.
- إذاً، لقد انتقل النظام من حالة حتمية إلى حالة جديدة تحتوي على **خيار لاحتمي** بين إجراءين **متوازيين**.

المرحلة الثالثة: تحليل الخيارات المستقبلية

الآن، ومن هذه الحالة الوسطى، تعرض الشريحة ثلاثة مسارات مستقبلية محتملة، وهي نفس فكرة الشريحة السابقة تماماً:

1. **إطلاق T2 أولاً** :نستهلك الرمز من P2 ونضع رمزاً في المكان اللاحق لـ T2 ويبقى T3 مُمكّناً وجاهزاً للإطلاق .
2. **إطلاق T3 أولاً** :نستهلك الرمز من P3 ونضع رمزاً في المكان اللاحق لـ T3 ويبقى T2 مُمكّناً وجاهزاً للإطلاق.
3. **إطلاق كليهما (T2 & T3)** :هذا يمثل إكمال كلا المسارين المتوازيين. النتيجة النهائية هي استهلاك الرمزين من P2 وP3 ، ووضع رمز في الأماكن اللاحقة لكل من T2 و T3

ملاحظة هامة :العبارة **"Any of the three states might be the next state"** تشير إلى الحالات المحتملة التي يمكن أن يصل إليها النظام بعد الحالة الوسطى، وليس بعد الحالة الابتدائية مباشرة.

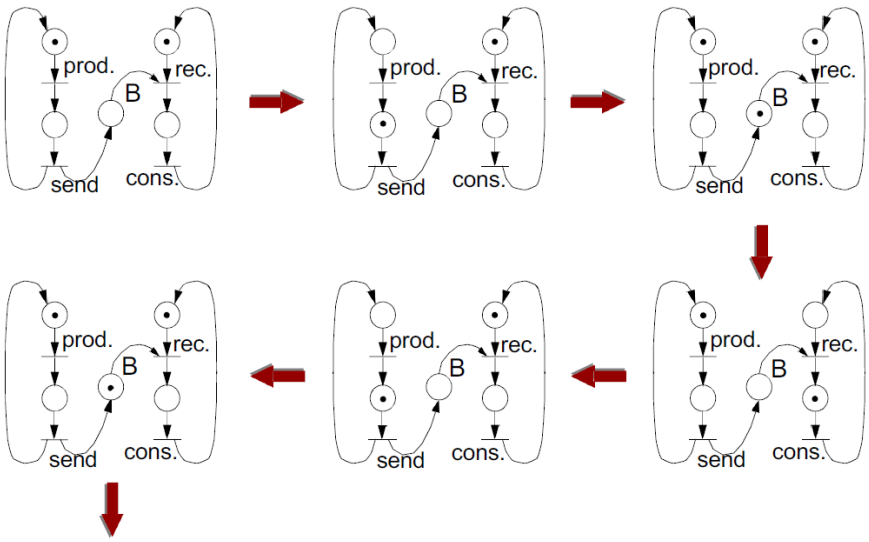
الخلاصة :

1. **البدء (T1) :** النظام يبدأ عملية محددة (حتمية).
2. **التوزيع :** هذه العملية الأولية تُفعِّل نظامين فرعيين مستقلين للعمل بالتوازي (مثل تفعيل نظام الاستشعار ونظام المحركات في نفس الوقت).
3. **التنفيذ المتوازي (T2, T3) :** يمكن لهذين النظامين الفرعيين القيام بمهامهما (مثل معايرة المستشعرات وفحص المحركات) بأي ترتيب، أو في نفس الوقت إذا كان العتاد يسمح بذلك.

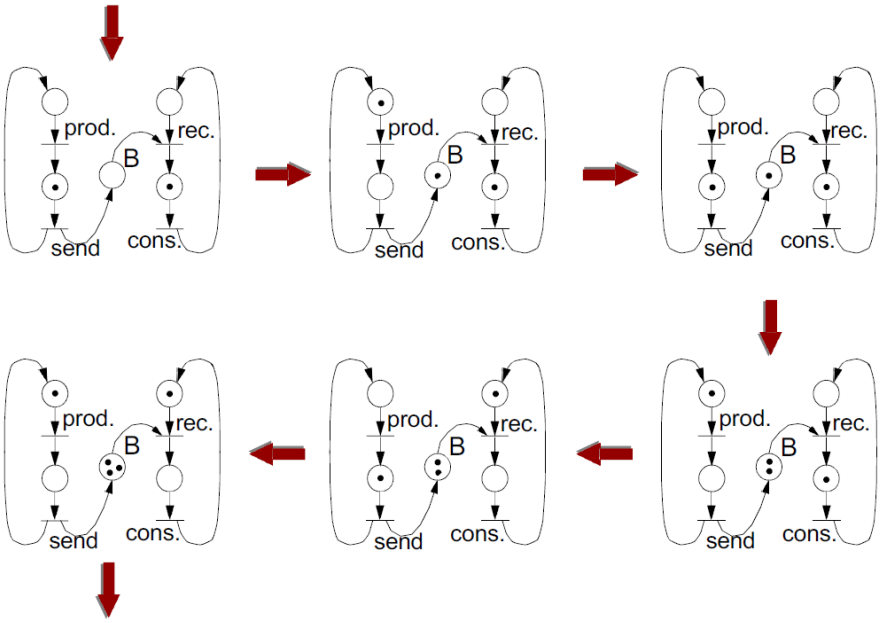
هذا النموذج يوضح كيف يمكن لشبكات بتري أن تصف تسلسل الإجراءات وتغير طبيعة سلوك النظام مع تطوره. إنها أداة قوية جداً لنمذجة مثل هذه السيناريوهات المعقدة التي نواجهها يومياً في هندسة الروبوت.

Petri Net Example

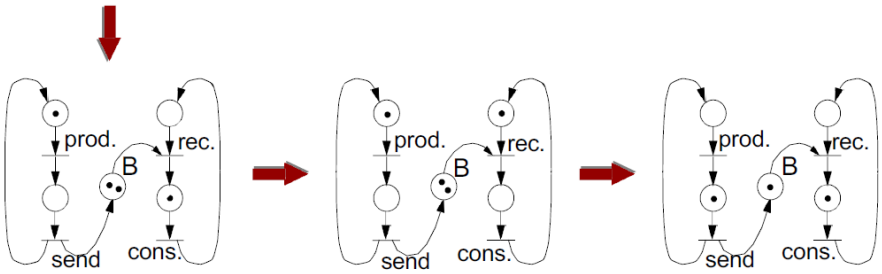
A producer and a consumer process communicating through a buffer:



Petri Net Example



Petri Net Example



- **Notice that the buffer is considered to be infinite (tokens accumulate in *B*).**

هذه الشرائح تأخذ كل المفاهيم النظرية التي تعلمناها وتطبقها على مثال عملي وكلاسيكي في عالم الحوسبة والأنظمة المدمجة: **مشكلة المنتج والمستهلك (The Producer-Consumer Problem).**

هذا المثال يوضح كيف يمكن لعمليتين مستقلتين (المنتج والمستهلك) أن تتواصلا وتتزامنا مع بعضهما البعض باستخدام مورد مشترك، وهو **المخزن المؤقت(Buffer)** .

تحليل مكونات النظام

أولاً سوف نفهم بنية شبكة بتري هذه. هي تتكون من جزأين رئيسيين يتواصلان عبر جزء ثالث:

1. **عملية المنتج (Producer Process) -على اليسار:**
   - هذه العملية تمثلها حلقة. مهمتها هي إنتاج بيانات أو عناصر (prod) ثم إرسالها (send) إلى المخزن المؤقت.
   - عندما تكون في الحالة العلوية (رمز في المكان العلوي الأيسر)، يكون المنتج "عاطلًا". عندما تنتج شيئًا، ينتقل الرمز إلى المكان السفلي الأيسر ("جاهز للإرسال"). بعد الإرسال، يعود الرمز إلى مكان "عاطل" لتبدأ الدورة من جديد.
2. **عملية المستهلك (Consumer Process) -على اليمين:**
   - هذه العملية هي أيضًا حلقة. مهمتها هي استقبال (rec) البيانات أو العناصر من المخزن المؤقت ثم استهلاكها(cons)
   - تشبه دورة المنتج: يبدأ المستهلك "عاطلًا" (رمز في المكان العلوي الأيمن)، ثم ينتقل إلى "جاهز للاستهلاك" (رمز في المكان السفلي الأيمن) بعد الاستلام، ثم يعود "عاطلًا" بعد الاستهلاك.
3. **المخزن المؤقت (Buffer) - المكان B في المنتصف:**
   - هذا المكان هو حلقة الوصل. إنه **المورد المشترك**.
   - عندما يُطلق المنتج انتقال send، يضع رمزًا في B
   - لكي يُطلق المستهلك انتقال rec، يجب أن يستهلك رمزًا من B
   - وجود رمز في B يعني "هناك عنصر في المخزن ينتظر الاستهلاك"

تتبع دورة عمل النظام

تُظهر لنا الشريحة تسلسل الأحداث، في النظام:

- **الحالة 1 (أعلى اليسار- البداية):** المنتج والمستهلك كلاهما عاطل. المخزن B فارغ. الانتقالات الممكنة(prod)
- **الحالة 2 (بعد إطلاق prod):** المنتج أنتج عنصرًا وهو الآن جاهز لإرساله.
- **الحالة 3 (بعد إطلاق send):** المنتج أرسل العنصر. نلاحظ أن رمزًا ظهر في المخزن B، وعاد المنتج إلى حالة الخمول، **جاهزًا لإنتاج عنصر جديد**.
- **الحالة 4 (بعد إطلاق rec):** في الحالة السابقة، رأى المستهلك أن هناك عنصرًا في المخزنB (به رمز)، فقام باستلامه. نلاحظ أن B أصبح فارغًا مرة أخرى. المستهلك الآن في حالة "جاهز للاستهلاك"
- **الحالة 5 (بعد إطلاق cons):** المستهلك استهلك العنصر وعاد إلى حالة الخمول، **جاهزًا لاستقبال عنصر جديد**.
- **الحالة 6 وما بعدها:** يستمر النظام في هذه الدورة. في الحالة الأخيرة المعروضة، نرى أن المنتج قد بدأ دورة إنتاج جديدة بينما المستهلك ينتظر.

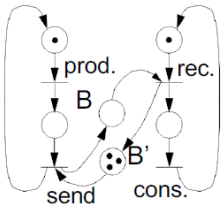
ربط المثال بالمفاهيم التي تعلمناها:

- **التوازي (Concurrency)** : بالنظر إلى **الحالة 3** .في هذه اللحظة، كان انتقال prod (إنتاج جديد) وانتقال rec(استقبال العنصر) كلاهما مُمكَّنًا. هذا يعني أن المنتج يمكن أن يبدأ في إنتاج العنصر التالي **في نفس الوقت** الذي يستعد فيه المستهلك لاستلام العنصر الحالي. هذا هو التوازي الحقيقي.
- **اللاحتمية (Nondeterminism)** : في نفس **الحالة 3**، كان لدى النظام خيار بين إطلاق prod أو rec. النموذج لا يفرض أيهما يحدث أولاً.
- **التزامن (Synchronization)** :لا يمكن للمستهلك أن يستلم (rec) إلا إذا كان المنتج قد أرسل (send) هذا التزامن يتم فرضه بواسطة المكان B هذا يمنع المستهلك من محاولة استهلاك بيانات غير موجودة.

في هذا النموذج، المخزن B يمكن أن يحتوي على عدد لا نهائي من الرموز (أي أن المنتج يمكن أن ينتج أسرع من المستهلك إلى ما لا نهاية). كيف يمكننا تعديل الشبكة لنمذجة **مخزن مؤقت بحجم محدود**، لنقل 5 عناصر فقط؟ هذا هو نوع المشاكل التي تساعدنا شبكات بتري في حلها قبل كتابة سطر واحد من الكود.

Petri Net Example

Here we have the same model as on the previous slides, but with limited buffer.
The buffer size is three (number of initial tokens in *B'*)

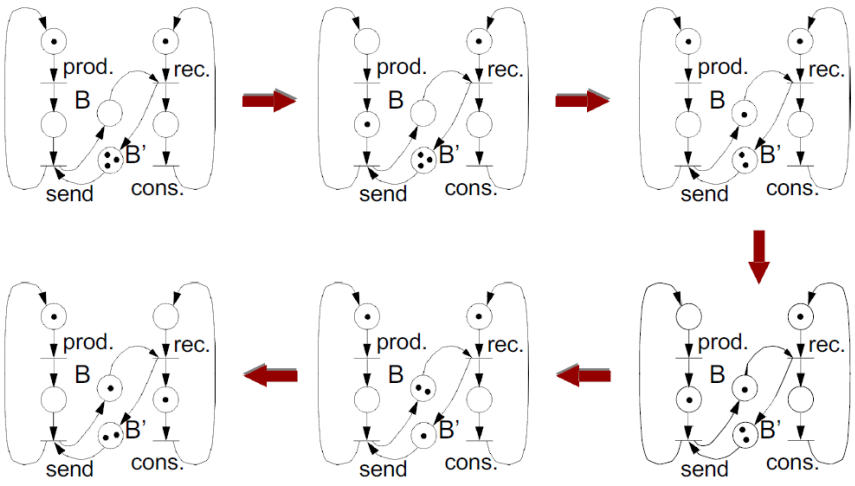


- **Nr. of tokens in B': how many free slots are available in the buffer;**
- **Nr. of tokens in B: how many messages (tokens) are in the buffer.**

  Total number of tokens in *B* and *B'* is constant (= 3).

Petri Net Example

Here we have the same model as on the previous slides, but with limited buffer.
The buffer size is three (number of initial tokens in *B'*)



Some Features and Applications of Petri Nets

- **Intuitive.**
  Easy to express concurrency, synchronisation, nondeterminism.

  Nondeterminism is an important difference between Petri nets and dataflow!

- **As an uninterpreted model, Petri Nets can be used for several, very different classes of problems.**
  - ***Uninterpreted model*: nothing has to be specified related to the particular activities associated to the transitions.**

شبكات بتري (Petri Nets) "هي نموذج رياضي لتمثيل الأنظمة المتزامنة(concurrent systems) ، وهي بالغة الأهمية في فهم وتصميم سلوك الروبوتات والأنظمة المدمجة بشكل عام. الميزات التي تجعلها متميزة:

1. **بديهية (Intuitive)**: اذا أردنا تمثيل تدفق العمليات داخل روبوت. "شبكات بتري" توفر لنا تمثيلاً رسوميًا بسيطًا وواضحًا. هي تتكون من دوائر (أماكنplaces) تمثل حالات أو موارد، ومستطيلات (انتقالات transitions) تمثل أحداثًا أو أفعالاً. السهام تربط بينها لتوضح تدفق التحكم أو البيانات. هذا الوضوح يساعدنا على تصور سلوك نظامنا بسهولة.
2. **سهولة التعبير عن التزامن، المزامنة، واللاحتمية:(Concurrency, Synchronization, and Nondeterminism)**

- **التزامن (Concurrency) :** في الروبوتات، غالبًا ما تحدث عدة عمليات في نفس الوقت. ذراع الروبوت يتحرك بينما مستشعراته تجمع البيانات، وفي نفس الوقت المعالج يقوم بعمليات حسابية. "شبكات بتري" تتيح لنا نمذجة هذه الأنشطة المتوازية بوضوح.

- **المزامنة (Synchronization) :** أحيانًا، يجب أن تنتظر عملية ما حتى تكتمل عملية أخرى قبل أن تبدأ. مثلاً، يجب أن يتوقف المحرك قبل أن يبدأ الذراع في الإمساك بشيء. "شبكات بتري" توفر آليات قوية لنمذجة هذه التبعيات والمزامنة بين المهام المختلفة.

- **اللاحتمية (Nondeterminism)** : وهذه نقطة جوهرية ومختلفة جدًا عن نماذج أخرى كنا قد تعرفنا عليها مثل "تدفق البيانات (Dataflow) ". في الأنظمة الحقيقية، خاصة في بيئات الروبوتات غير المنظمة، قد يكون هناك أكثر من مسار ممكن للسلوك التالي للنظام. على سبيل المثال، قد يكون هناك أكثر من خيار للروبوت ليقوم به في لحظة معينة بناءً على المدخلات البيئية أو حتى عوامل داخلية لا يمكن تحديدها مسبقًا بدقة (مثل توقيت وصول الإشارة). "شبكات بتري" تسمح لنا بنمذجة هذه الاحتمالات المتعددة، مما يعكس واقع الأنظمة المدمجة المعقدة. هذه القدرة على التعامل مع اللاحتمية تجعلها أداة قوية لتحليل الأخطاء المحتملة أو السلوكيات غير المتوقعة في تصميماتنا.

3. **كنموذج غير مفسر (Uninterpreted Model) ، يمكن استخدام شبكات بتري لعدة فئات مختلفة جدًا من المشاكل:**

- ماذا يعني "غير مفسر"؟ ببساطة، عندما نقوم ببناء نموذج ، **نحن لا نحتاج لتحديد التفاصيل الدقيقة للأنشطة المرتبطة بكل "انتقال (transition)"**. لا يهم ما إذا كانت الحركة هي "تحريك الذراع"، أو "قراءة حساس"، أو "حساب مسار". ما يهم هو العلاقات السببية والزمنية (causal and temporal relationships) بين هذه الأنشطة. هذا يعني أن نفس نموذج شبكة بتري يمكن أن يمثل سلوك روبوت، أو بروتوكول شبكة اتصالات، أو حتى عملية تصنيع في مصنع. هذا يجعلها أداة تحليل وتصميم مرنة للغاية يمكننا تطبيقها على شتى أنواع المشاكل في الأنظمة المدمجة.

باختصار، "شبكات بتري" تمنحنا القدرة على:

- تصور سلوك أنظمتنا المعقدة بوضوح.
- التحكم في التزامن والمزامنة بين مهام الروبوت المختلفة.
- فهم وتحليل السلوكيات المحتملة في ظل عدم اليقين (اللاحتمية).
- تطبيق نفس الأداة لتحليل مشاكل متنوعة دون الحاجة لإعادة تعريف الأنشطة الأساسية.

هذه الميزات تجعلها حجر الزاوية في تصميم وتحليل الأنظمة المدمجة الموثوقة والفعالة

Some Features and Applications of Petri Nets

- **Petri Nets have been intensively used for modeling and analysis of industrial production systems, information systems, but also**
  - **Computer architectures**
  - **Operating systems**
  - **Concurrent programs**
  - **Distributed systems**
  - **Hardware systems**

المجالات الواسعة التي تم فيها استخدام شبكات بتري بشكل مكثف والتي تشمل:

- **معماريات الحاسوب (Computer architectures)** : أي كيفية بناء وتصميم المكونات المختلفة للحاسوب وكيف تتفاعل مع بعضها. يمكن لشبكات بتري تحليل تدفق البيانات والتحكم داخل المعالجات أو الذاكرة.
- **أنظمة التشغيل (Operating systems)** : كيف تقوم أنظمة التشغيل بإدارة الموارد، جدولة العمليات، والتعامل مع التزامن بين المهام المختلفة.
- **البرامج المتزامنة (Concurrent programs)** : البرامج التي تنفذ أجزاء متعددة منها في نفس الوقت، مثل التطبيقات متعددة المهام(multithreaded applications)
- **الأنظمة الموزعة (Distributed systems)** : الأنظمة التي تتكون من عدة أجهزة حاسوب منفصلة تتواصل وتعمل معًا لتحقيق هدف مشترك، مثل الحوسبة السحابية أو شبكات الحساسات.
- **Hardware systems** : تصميم وتحليل الدرات الإلكترونية والأنظمة المادية التي تتفاعل بشكل متزامن.

هذه الشريحة تؤكد على **الشمولية الكبيرة لشبكات بتري كأداة تحليل ونمذجة** .إنها ليست مقتصرة على مجال واحد، بل يمكن تطبيقها في أي نظام تتواجد فيه عمليات متزامنة، تفاعلات معقدة، أو حالات تتطلب تزامنًا دقيقًا.

Properties and Analysis of Petri Nets

- **Several properties of the system can be analysed using Petri nets:**
  - ***Boundedness*: number of tokens in a place does not exceed a limit.**
    If this limit is 1, the property is sometimes called *safeness*.
    - **You can check that available resources are not exceeded.**
  - ***Liveness*: A transition *t* is called live if for every possible marking there exists a chance for that transition to become enabled.**
    The whole net is live, if all its transitions are live.
    - **Important in order to check that the system is not deadlocked.**
  - ***Reachability*: given a current marking M and another marking M', does there exist a sequence of transitions by which M' can be obtained?**
    - **You can check that a certain desired state (marking) is reached.**
    - **You can check that a certain undesired state is never reached.**

هذه الشريحة تتناول كيف يمكننا استخدام نماذج "شبكات بتري" ليس فقط لتمثيل الأنظمة، بل أيضًا **لتحليل سلوكيات معينة والتأكد من أنها تتصرف بالشكل المطلوب أو أنها لا تصل إلى حالات غير مرغوبة.**

هنا ثلاث خصائص رئيسية يمكن تحليلها باستخدام شبكات بتري:

1. **الحدودية (Boundedness):**
   - تعني أن "عدد التوكنز (tokens) في أي مكان (place) لا يتجاوز حدًا معينًا". التوكنز هنا تمثل الموارد أو البيانات أو الحالات.
   - **Safeness (السلامة):** إذا كان هذا الحد هو 1 (أي لا يمكن أن يكون هناك أكثر من توكن واحد في المكان)، فإن هذه الخاصية تسمى أحيانًا "السلامة". هذا يعني أن المكان لا يمكن أن يحتوي على أكثر من "توكن" واحد في أي وقت.
   - **التطبيق العملي:** "يمكننا التحقق من أن الموارد المتاحة لا يتم تجاوزها". لنتخيل أن لدينا ذاكرة تخزين مؤقتة (buffer) في روبوت، باستخدام الحدودية، يمكننا التأكد من أن الروبوت لن يحاول تخزين بيانات أكثر من سعة الذاكرة، مما يمنع تجاوز الموارد ويحافظ على استقرار النظام.
2. **الحيوية (Liveness):**
   - يقال عن انتقال (transition) أنه حي (live) إذا كان لكل تأشير (marking) ممكن (أي كل حالة ممكنة للنظام) هناك فرصة لأن يصبح هذا الانتقال مُفعلًا (enabled). بمعنى آخر، لن يتم حظر هذا الانتقال للأبد.
   - "الشبكة بأكملها تكون 'حية' إذا كانت جميع انتقالاتها 'حية'."

- **التطبيق العملي:** مهمة للتحقق من أن النظام لا يتعرض للتعطل (deadlock). هذه نقطة حاسمة جدًا في تصميم الأنظمة المدمجة والروبوتات. الـ"deadlock" يعني أن النظام يصل إلى حالة لا يمكنه فيها التقدم، حيث تنتظر العمليات بعضها البعض إلى الأبد. مثال: ذراعين للروبوت يحاولان الوصول إلى نفس المورد في نفس الوقت ويحتاج كل منهما إلى ما يحمله الآخر. تحليل "الحيوية" يسمح لنا بالتأكد من أن نظامنا لن يدخل في مثل هذه الحالات التي يتوقف فيها عن العمل بشكل كامل.

3. **قابلية الوصول (Reachability):**

- بالنظر إلى تأشير (marking) حالي M (حالة حالية للنظام) وتأشير آخر M' (حالة مستهدفة)، هل توجد سلسلة من الانتقالات يمكن من خلالها الوصول إلى M'؟

- **التطبيقات العملية:**

  - يمكننا التحقق من الوصول إلى حالة مرغوبة معينة. مثلاً، هل يمكن لروبوت أن يصل إلى نقطة معينة في المساحة؟ أو هل يمكن أن ينهي مهمة معينة بنجاح؟
  - يمكننا التحقق من عدم الوصول أبدًا إلى حالة غير مرغوبة معينة. وهذه بنفس الأهمية. هل يمكن لروبوت أن يصل إلى حالة تصادم؟ أو هل يمكن أن يدخل في حلقة لا نهائية من الأخطاء؟ باستخدام قابلية الوصول، يمكننا التأكد من أن التصميم الخاص بنا يتجنب هذه الحالات الخطيرة أو غير المرغوبة.

باختصار، هذه الخصائص توفر لنا أدوات تحليل قوية لشبكات بتري:

- **ضمان سلامة الموارد (Safety of Resources).**
- **منع توقف النظام (Preventing Deadlocks).**
- **التحقق من تحقيق الأهداف وتجنب الأخطاء (Verifying Goals and Avoiding Errors).**

هذه القدرات التحليلية تجعل "شبكات بتري" ليست مجرد أداة للنمذجة، بل أداة فعالة جدًا للتحقق من صحة وقوة تصاميمنا للأنظمة المدمجة والروبوتات.

Extended Petri Net Models

Basic Petri Net models have a limited expressive power.

- **Timed Petri Nets**
  - **Transitions have associated times (time intervals)**
  - **Tokens are carrying time stamps.**

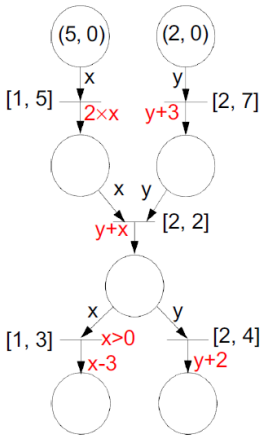
  With timed Petri nets we can model the timing aspects

- **Coloured Petri Nets**
  - **Tokens have associated values**
  - **Transitions have associated functions**

  Coloured Petri Nets are similar to dataflow models (but also capture nondeterminism!).

Extended Petri Net Models

Coloured and Timed Petri net



- **Tokens carry Time stamps**
- **Transitions have associated time (interval)**

- **Tokens have associated values**
- **Transitions have associated functions and guards**

نماذج شبكات بتري الأساسية لديها قدرة تعبيرية محدودة .هذا يعني أن النموذج الأساسي، على الرغم من قوته في نمذجة التزامن و اللاحتمية، قد لا يكون كافيًا لالتقاط جميع تفاصيل الأنظمة المدمجة المعقدة، خاصة تلك التي تعتمد على الزمن أو أنواع البيانات المختلفة. لذا، تم تطوير توسعات لـ"شبكات بتري" لزيادة قدرتها التعبيرية:

1. **شبكات بتري الزمنية(Timed Petri Nets) :**

- في الأنظمة المدمجة والروبوتات، التوقيت (timing) حاسم جدًا. ليس فقط تسلسل الأحداث، بل أيضًا متى تحدث هذه الأحداث، أو كم تستغرق من الوقت. هذا هو ما تعالجه "شبكات بتري الزمنية"

- **كيف تعمل؟**
  - "**الانتقالات (Transitions) لديها فترات زمنية مرتبطة بها** ".هذا يعني أن كل عملية أو حدث (يمثله الانتقال) يستغرق زمناً معينًا لإكماله، أو يجب أن ينتظر فترة زمنية محددة قبل أن يتفعل.
  - "**التوكنز (Tokens) تحمل طو ابع زمنية (time stamps)** " . التوكن لم يعد مجرد علامة، بل يحمل معه معلومات عن الزمن الذي وصل فيه إلى مكان معين.
- **مع شبكات بتري الزمنية، يمكننا نمذجة الجو انب الزمنية** . هذا يمكننا من تحليل الأداء الزمني لنظام الروبوت الخاص بنا، مثل التأكد من أن مهمة معينة ستكتمل في إطار زمني محدد، أو تحليل فترات التأخير (latencies) في الاستجابة. وهذا أمر حيوي في أنظمة التحكم في الزمن الحقيقي (real-time control systems) التي نعمل عليها.

2. **شبكات بتري الملونة(Coloured Petri Nets) :**

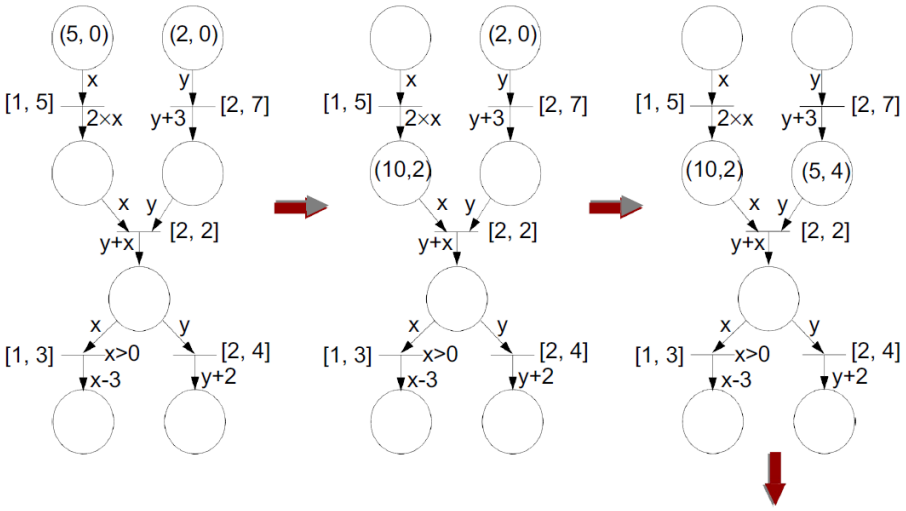
- النموذج الأساسي لشبكات بتري يتعامل مع التوكنز ككيانات متطابقة. لكن في الأنظمة الحقيقية، قد تكون التوكنز تمثل أنواعًا مختلفة من البيانات أو كائنات مختلفة.
- **كيف تعمل؟**
  - "**التوكنزلديها قيم مرتبطة بها** .بدلاً من مجرد كونها "توكن"، يمكن أن يكون التوكن "توكن بيانات رقمية"، أو "توكن يمثل مفصل الروبوت الأول"، أو "توكن يمثل خطأ في حساس معين". كل توكن له "لون" أو "قيمة" تميزه.
  - "**الانتقالات (Transitions) لديها تو ابع (functions) مرتبطة بها** ".لم يعد الانتقال مجرد شرط لمرور التوكنز، بل يمكن أن يقوم بمعالجة أو تغيير قيم التوكنز التي تمر عبره. على سبيل المثال، قد يقوم الانتقال بجمع قيم توكنين، أو تغيير حالة توكن معين بناءً على تابع محدد.
- **العلاقة بـ"تدفق البيانات" و اللاحتمية" ::شبكات بتري الملونة تشبه نماذج تدفق البيانات (dataflow models) ولكنها أيضًا تلتقط اللاحتمية** .هذا يعني أنها قوية في نمذجة تدفق البيانات ومعالجتها (مثل أنظمة تدفق البيانات)، ولكنها تحتفظ أيضًا بقدرتها الأساسية على نمذجة اللاحتمية، وهو ما يميزها عن نماذج تدفق البيانات البحتة. هذه المرونة تجعلها مثالية لنمذجة سلوكيات الروبوتات المعقدة التي تتعامل مع أنواع مختلفة من البيانات وتتخذ قرارات بناءً على تلك البيانات.

باختصار، هذه النماذج الموسعة لـ"شبكات بتري" تمنحنا أدوات أقوى لـ:

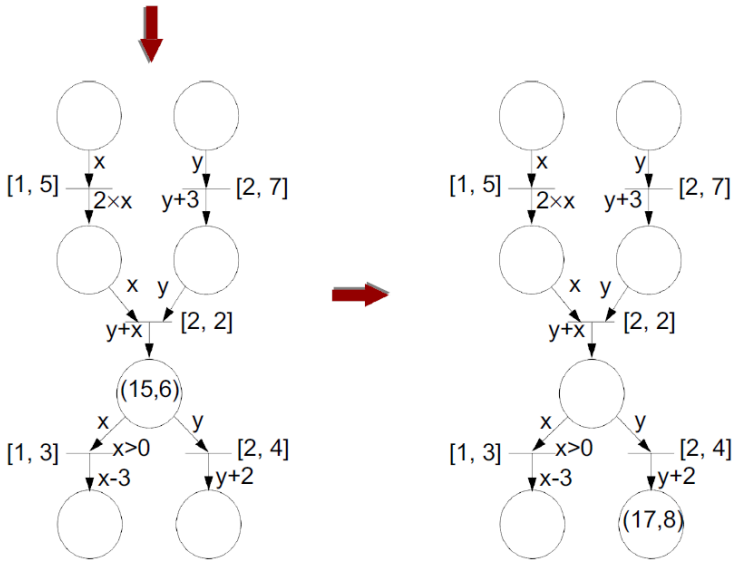
- **التعامل مع الجو انب الزمنية** (Timed Petri Nets) في تصميم نظام الروبوت، مما يضمن أداءً موثوقًا وفي الوقت المناسب.
- **نمذجة تدفق البيانات ومعالجتها بشكل أكثر تفصيلاً** (Coloured Petri Nets)، مع الحفاظ على قدرة الشبكات الأساسية على التعامل مع التزامن و اللاحتمية، مما يسمح بنمذجة سلوكيات أكثر تعقيدًا للروبوتات.

هذه الامتدادات تجعل "شبكات بتري" أداة شاملة وقوية بشكل استثنائي لتصميم وتحليل الأنظمة المدمجة .

Extended Petri Net Models



Extended Petri Net Models



سوف نوضح بالخطوات كيفية عمل نموذج "شبكة بتري الملونة والزمنية (Coloured and Timed Petri Net) "بشكل ديناميكي، وكيف تتفاعل التوكنز وتتغير قيمها وأزمنتها مع تقدم النظام.

لنتتبع تسلسل الأحداث خطوة بخطوة:

الحالة الابتدائية (الرسم الأقصى على اليسار في الشريحة الأولى):

- لدينا توكنان في الأعلى :
  - في المكان العلوي الأيسر: توكن بقيمة 5وطابع زمني 0 نشير إليه بـ (5, 0)
  - في المكان العلوي الأيمن: توكن بقيمة 2وطابع زمني 0 نشير إليه بـ (2, 0)
- نلاحظ أن كل انتقال له :
  - **فترة زمنية (Time Interval) :**مثل [1, 5]أو [2, 7]، تحدد كم يستغرق الانتقال لإكماله.
  - **تو ابع (Functions) :** مثل 2xxأو y+3، تحدد كيفية تغيير قيم التوكنز.
  - **قيود :**مثل x>0، وهي شروط يجب أن تتحقق لكي يتم تفعيل الانتقال.

الخطوة الأولى (الرسم الأوسط في الشريحة الأولى):

- تم تفعيل الانتقال العلوي الأيسر (المرتبط بـ x).
  - كان لديه توكن (5, 0)
  - التابع المرتبط به هو 2xx، لذا قيمة التوكن الجديد ستكون 2 * 5 = 10
  - هذا الانتقال له فترة زمنية [1, 5]بناءً على النموذج، أي أن الانتقال استغرق وحدتين زمنيتين، لذا الطابع الزمني للتوكن الجديد يصبح 0 + 2 = 2
  - النتيجة: توكن (10, 2)ينتقل إلى المكان الأوسط الأيسر.
- الجزء العلوي الأيمن من الشبكة لا يزال كما هو، حيث أن الانتقال الأيمن لم يتفعل بعد.

الخطوة الثانية (الرسم الأقصى على اليمين في الشريحة الأولى):

- تم تفعيل الانتقال العلوي الأيمن (المرتبط بـ y)
  - كان لديه توكن (2, 0)
  - التابع المرتبط به هو y+3، لذا قيمة التوكن الجديد ستكون 2 + 3 = 5
  - هذا الانتقال له فترة زمنية [2, 7]، أي أن الانتقال استغرق 4 وحدات زمنية (ضمن هذا النطاق)، لذا الطابع الزمني للتوكن الجديد يصبح 0 + 4 = 4
  - النتيجة: توكن (5, 4)ينتقل إلى المكان الأوسط الأيمن.
- المكان الأوسط الأيسر لا يزال يحتوي على التوكن (10, 2)

الخطوة الثالثة (الرسم الأقصى على اليسار في الشريحة الثانية - استكمالاً للشريحة الأولى):

- بما أن المكانين الأوسطين (الأيسر والأيمن) أصبحا يحتويان على توكنز (10, 2) و (5, 4)على التوالي، فإن الانتقال الأوسط السفلي (المرتبط بـ x و y) أصبح قابلاً للتفعيل .

- هذا الانتقال يتطلب توكن x وتوكن y
- التابع المرتبط به هو y+x، لذا قيمة التوكن الناتج ستكون 15 = 10 + 5
- هذا الانتقال له فترة زمنية [2, 2]، أي أنه يستغرق بالضبط وحدتين زمنيتين. ولتحديد الطابع الزمني للتوكن الناتج، نأخذ أقصى طابع زمني من التوكنز الداخلة ونضيف عليه زمن الانتقال:6 = 2 + 4 = 2 + Max(2, 4)
- النتيجة: توكن (15, 6)ينتقل إلى المكان الأوسط السفلي.

- يمكن ملاحظة أن الأماكن العلوية الآن فارغة، والأماكن الوسطى التي كانت تحمل التوكنز (10,2) و (5,4) أصبحت فارغة.

الخطوة الرابعة (الرسم الأقصى على اليمين في الشريحة الثانية):

- الآن، لدينا توكن (15, 6)في المكان الأوسط السفلي. هذا يجعل الانتقال السفلي الأيمن (المرتبط بـ y) قابلاً للتفعيل.
- نلاحظ أن الانتقال السفلي الأيسر له قيد x>0 و تابع x-3 لكنه لم يتفعل في هذه الخطوة (ربما لم يختار النظام هذا المسار، أو لم يكن هناك توكن x بالتعريف المناسب له).
- الانتقال السفلي الأيمن :
  - استقبل توكن (15, 6)
  - التابع المرتبط به هو y+2، لذا قيمة التوكن الجديد ستكون 17 = 2 + 15
  - هذا الانتقال له فترة زمنية [2, 4]. أي أنه استغرق وحدتين زمنيتين (ضمن النطاق)، لذا الطابع الزمني للتوكن الجديد يصبح 8 = 2 + 6
  - النتيجة: توكن (17, 8)ينتقل إلى المكان السفلي الأيمن.
- المكان الأوسط السفلي يصبح فارغًا.

أهمية هذا المثال:

هذا التسلسل يوضح بشكل ممتاز كيف يمكن لـ"شبكات بتري الملونة والزمنية" أن تمثل سلوكًا ديناميكيًا معقدًا:

- **تدفق البيانات :**التوكنز تحمل قيما (5 ، 2، ثم 10، 5، ثم 15، ثم 17).
- **معالجة البيانات :**التوابع (2xx، y+3، y+x، y+2) تقوم بتحويل هذه القيم.
- **القيود :** x>0 تتحكم في مسار تدفق البيانات.
- **الجوانب الزمنية :**الطوابع الزمنية(0 ، ثم 2، 4، ثم 6، ثم 8) وفترات زمن الانتقال([1, 5] ، [2, 7]، [2, 2]، [2, 4]) تسمح بتتبع التوقيت الدقيق للأحداث.

هذا النوع من النمذجة مهم جداً في تصميم الروبوتات حيث يجب أن تعمل الأنظمة الفرعية المختلفة معًا، وتتعامل مع بيانات متنوعة، وتستجيب في أطر زمنية محددة. إنه يسمح لنا بتحليل أداء النظام، وتحديد الاختناقات المحتملة، والتأكد من صحة السلوك في بيئات حقيقية.

Extended Petri Net Models

- **Extended Petri Nets have a larger expressive power then classical Petri Nets.**

Analysis is more complex; the formal analysis of properties can take very large amounts of time (memory).

- **Simulation of the Petri Net is very often used in order to verify the system and to estimate performance**

- **"شبكات بتري الموسعة لديها قدرة تعبيرية أكبر من شبكات بتري التقليدية ".**
  - هذا ما رأيناه في الشريحة السابقة عندما تحدثنا عن "شبكات بتري الزمنية" التي تتعامل مع الزمن، و"شبكات بتري الملونة" التي تتعامل مع أنواع مختلفة من البيانات وتتيح توابع ومعالجة للقيم. هذه الإضافات تجعل النموذج أقرب بكثير لتعقيد الأنظمة المدمجة والروبوتات في العالم الحقيقي.

ولكن، مع هذه القوة التعبيرية تأتي بعض التحديات:

- **"التحليل يصبح أكثر تعقيدًا؛ التحليل الرسمي للخصائص يمكن أن يستغرق كميات كبيرة جدًا من الوقت (والذاكرة) ".**
  - عندما تحدثنا عن "التحقق الرسمي" لخصائص مثل الحدودية (Boundedness) والحيوية (Liveness) وقابلية الوصول(Reachability) ، في الشبكات التقليدية، يمكن القيام بذلك باستخدام أدوات رياضية. لكن عندما نضيف الألوان (البيانات والقيم المتغيرة) والتوقيتات (الأبعاد الزمنية)، يصبح "فضاء الحالات (state space) "للنظام أكبر بكثير ومعقدًا جدًا. هذا يعني أن الأدوات التحليلية الرسمية قد تحتاج إلى كميات هائلة من الوقت والذاكرة الحاسوبية لإجراء هذه التحققات بشكل كامل، وقد يصبح الأمر غير عملي لأنظمة الروبوتات المعقدة.

إذاً، ما هو الحل البديل أو المكمل الذي غالبًا ما نلجأ إليه؟

- **"غالبًا ما تُستخدم محاكاة شبكة بتري للتحقق من النظام وتقدير الأداء ".**
  - نظرًا لصعوبة التحليل الرسمي في بعض الحالات المعقدة، فإننا نلجأ إلى **المحاكاة (Simulation)** . بدلاً من محاولة إثبات كل خاصية رياضيًا لجميع الحالات الممكنة، نقوم بـ"تشغيل" النموذج لعدد كبير من السيناريوهات والمدخلات المحتملة.
  - **للتحقق من النظام (Verify the system)** :من خلال المحاكاة، يمكننا رؤية كيف يتصرف الروبوت أو النظام المدمج تحت ظروف مختلفة، والتأكد من أنه يعمل بشكل صحيح ويتجنب الحالات غير المرغوبة.

- **لتقدير الأداء (Estimate performance)** :يمكننا قياس أزمنة الاستجابة، استخدام الموارد، ومعدلات الإنتاجية، وهي معلومات حاسمة لتقييم كفاءة تصميم نظام الروبوت الخاص بنا.

الخلاصة:" شبكات بتري الموسعة" هي أداة قوية للغاية لنمذجة تعقيد الأنظمة المدمجة، ولكن يجب أن نكون على دراية بأن التحليل الرسمي لها قد يكون مكلفًا من الناحية الحاسوبية. لذلك، فإن **المحاكاة** تصبح أداة لا غنى عنها ، حيث تمكننا من اختبار وتصحيح وتطوير أنظمتنا بشكل فعال حتى عندما يكون التحليل الرسمي صعبًا أو مستحيلاً.