Explosion combinatoire.

Problématique

- On veut vérifier automatiquement:

$$\mathcal{M} \models^{?} \phi.$$

- Étapes:
  1. On modélise $\mathcal{M}$.
  2. On modélise / spécifie la propriété ϕ.
  3. On vérifie si $\mathcal{M} \models^{?} p$ avec un certain algorithme ou une certaine méthode.

Modélisation du système $\mathcal{M}$

- Jusqu'à maintenant dans ce cours, on a dit qu'on utilisait un système de transitions ou une structure de Kripke pour modéliser $\mathcal{M}$.
- Mais en pratique, on utilise plutôt un *langage de modélisation*.

Exemples de langages de modélisation:

- Produit $\left\{ \begin{array}{l} \text{d'automates} \\ \text{de textes (programmes)} \end{array} \right.$ à

  $\left\{ \begin{array}{l} \text{variables partagées} \\ \text{échanges de messages} \end{array} \right\}$

- Réseaux de Pétri.
- Algèbres de Processus (CSP, CCS,...).
- LOTOS, LUSTRE, ESTEREL ...
- Descriptions logiques ...
- Langage C, Pascal ...
- ...

Exemple 0: n processus en parallèle

- n-copies du processus simple vu précédemment, alternant entre NC et SC, qui évoluent en parallèle de façon asynchrone.

- Structure de Kripke sur $AP = \{crit_1, \ldots, crit_n\}$
  - $S = \{NC, SC\}^n$
  - $S_0 = (NC, \ldots, NC)$
  - $R = \{(x, x') | \exists i (x_i \neq x'_i \& x_j = x'_j$ pour $j \neq i\}$
  - $L(x) = \cup_{x_i = SC} crit_i$

- Description de taille $O(n^2)$ d'un espace de taille 2^n.

Exemple 1: langage de Manna Pnueli

Instructions:

1. $x := e$
2. $skip$
3. $P_1; P_2$
4. $if\ b\ then\ P_1\ else\ P_2$
5. $while\ b\ do\ P_1$
6. $P_1 \| P_2$.

Algorithme simple d'exclusion mutue

$$P_1 :: \left[\begin{array}{ll} \bot : & \\ l_0 : & \mathit{while}\ \mathit{true}\ \mathit{do} \\ & \left[\begin{array}{ll} NC_0 : & \mathit{wait}(\mathit{turn} = 0); \\ CR_0 : & \mathit{turn} := 1; \end{array} \right] \end{array} \right]$$

$$\|$$

$$P_2 :: \left[\begin{array}{ll} \bot : & \\ l_1 : & \mathit{while}\ \mathit{true}\ \mathit{do} \\ & \left[\begin{array}{ll} NC_1 : & \mathit{wait}(\mathit{turn} = 1); \\ CR_1 : & \mathit{turn} := 0; \end{array} \right] \end{array} \right]$$

Exemple: modélisation de l'algorithm

$$P_1 :: \left[\begin{array}{ll} l_0 : & while\ true\ do \\ & \left[\begin{array}{ll} out : & (y_1, x) := (True, 1) \\ req : & wait\ (\neg y_2) \vee (x = 2) \\ in : & < critical > \\ exit : & y_1 := False \end{array} \right] \end{array} \right]$$

$$\|$$

$$P_2 :: \left[\begin{array}{ll} l_0 : & while\ true\ do \\ & \left[\begin{array}{ll} out : & (y_2, x) := (True, 2) \\ req : & wait\ (\neg y_1) \vee (x = 1) \\ in : & < critical > \\ exit : & y_2 := False \end{array} \right] \end{array} \right]$$

Exemple 2: langage graphique

- Exemple: modélisation d'un compteur modulo 8.

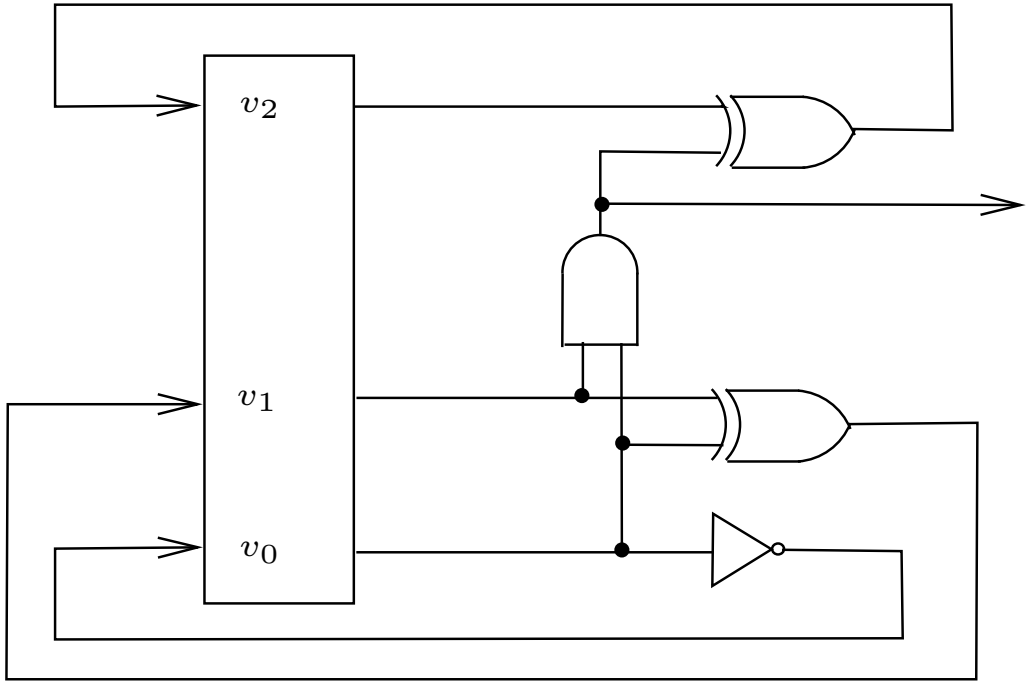

Figure 1: Modélisation d'un compteur modulo 8

$$
\begin{array}{lcll}
P_0 & \equiv & (v_0' & = \quad \neg v_0) \\
P_1 & \equiv & (v_1' & = \quad v_0 \oplus v_1) \\
P_2 & \equiv & (v_2' & = \quad (v_0 \wedge v_1) \oplus v_2)
\end{array}
$$

Rappels: systèmes de transitions

- Un système de transitions (resp. une structure de Kripke) est $M = (S, S_0, R)$ (resp. $M = (S, S_0, R, L)$) avec
  1. S un ensemble d'états.
  2. $S_0 \subseteq S$ l'ensemble des états initiaux.
  3. $R \subseteq S \times S$ une relation de transition totale. i.e. pour tout s il existe t tel que $R(s, t)$.
  4. (resp. $L : S \to 2^{AP}$ une fonction d'étiquetage des états par les formules atomiques.)

- Un chemin de M débutant en s est une suite $\pi = s_0, s_1, \ldots$ telle que $s_0 = s$ et $R(s_i, s_{i+1})$ pour tout $i \geq 0$.
- Le chemin est initialisé si en outre $s \in S_0$.

Systèmes discrets

- But: distinguer les systèmes de transitions de leurs descriptions, i.e. les systèmes de transitions des systèmes discrets.
- Un *système discret* est $\mathcal{M} = (V, \mathcal{S}_0, \mathcal{R})$ où
  1. V est un ensemble fini de symboles de variables. Chaque variable $x \in V$ prend ses valeurs dans un domaine D_x fini.
  2. $\mathcal{S}_0$ est une assertion sur V.
     Une assertion est une combinaison booléenne de formules atomiques. Exemple: $x = 0 \wedge y = 1$.
  3. $\mathcal{R}$ est une assertion sur $V \times V'$.
     Lorsque $V = \{x_1, \ldots, x_d\}$, V' dénote $V' = \{x'_1, \ldots, x'_d\}$, i.e. la copie primée des variables de V. Exemple $\mathcal{R}$: $x = 0 \rightarrow x' = 1 \vee y = 0 \rightarrow x' = 0 \wedge y' = 1$.

Exemple. Exclusion Mutuelle Simple

- Variables: pc_1, pc_2 avec $pc_i : \{\bot, l_i, NC_i, CR_i\}$, $pc : \{m, \bot\}$ et $turn : \{0, 1\}$
- S: valuations des variables.
- $\mathcal{S}_0$: $pc_1 = \bot \wedge pc_2 = \bot$.
- $\mathcal{R}(pc_1, pc_2, turn)$ disjonction de

$$pc = m \wedge pc'_0 = l_0 \wedge pc'_1 = l_1 \wedge \ pc' = \bot$$
$$pc_i = l_i \ \wedge \ pc'_i = NC_i$$
$$pc_i = NC_i \ \wedge \ pc'_i = CR_i \wedge \ turn = i$$
$$pc_i = CR_i \ \wedge \ pc'_i = l_i \wedge \ turn' = 1 - turn$$
$$pc_i = NC_i \ \wedge \ pc'_i = NC_i \wedge \ turn \neq i$$

(Convention: dans les formules logiques de transitions, on oublie les $same$, c'est-à-dire on ne note pas les variables non-modifiées: par exemple $pc = m \wedge pc'_0 = l_0 \wedge pc'_1 = l_1 \wedge \ pc' = \bot$ se noterait s'il on oubliait pas la variable $turn$ $pc = m \wedge pc'_0 = l_0 \wedge pc'_1 = l_1 \wedge \ pc' = \bot \wedge turn' = turn$.).

Exemple. Peterson

- Variables: $pc_1, pc_2 : \{out, req, in, exit\}$, $y_1, y_2 : \{False, True\}$, $x : \{1, 2\}$.
- S: valuations des variables.
- $\mathcal{S}_0$: $pc_1 = out \wedge pc_2 = out \wedge y_1 = y_2 = False$.

- $\mathcal{R}(pc_1, pc_2, y_1, y_2, x)$ disjonction de

1. $pc_1 = out \rightarrow pc_1' = req \wedge x' = 1 \wedge y_1'$
2. $pc_1 = req \wedge \neg y_2 \rightarrow pc_1' = in$
3. $pc_1 = req \wedge x = 2 \rightarrow pc_1' = in$
4. $pc_1 = in \rightarrow pc_1' = exit$
5. $pc_1 = exit \rightarrow pc_1' = out \wedge \neg y_1'$
6. $pc_2 = out \rightarrow pc_2' = req \wedge x' = 2 \wedge y_2'$
7. $pc2 = req \wedge \neg y_1 \rightarrow pc_2' = in$
8. $pc_2 = req \wedge x = 1 \rightarrow pc_2' = in$
9. $pc_2 = in \rightarrow pc_2' = exit$
10. $pc_2 = exit \rightarrow pc_2' = out \wedge \neg y_2'$.

(On utilise ici la convention précédente).

Exemple. Compteur modulo 8

- $V = \{a, b, c\}$, $\mathcal{S}_I = \dashv = \prime \wedge \lfloor = \prime \wedge \rfloor = \prime$.
- $\mathcal{R}_0(V, V') : v'_0 = \neg v_0 \wedge same(v_1, v_2)$.
- $\mathcal{R}_1(V, V') : v'_1 = v_0 \oplus v_1 \wedge same(v_0, v_2)$.
- $\mathcal{R}_2(V, V') : v'_2 = ((v_0 \wedge v_1) \oplus v_2) \wedge same(v_0, v_1)$.

($same(v_1, \ldots, v_n)$ signifie $v'_1 = v_1 \wedge \ldots v'_n = v_n$).

- Composition synchrone: $\mathcal{R}(V, V') = \bigwedge_i \mathcal{R}_i(V, V')$.
- Composition asynchrone: $\mathcal{R}(V, V') = \bigvee_i \mathcal{R}_i(V, V')$.

Tailles

La taille

- d'un ensemble S, notée $|S|$, est le nombre d'éléments de cet ensemble.
- d'une assertion ϕ, notée $|\phi|$, est la taille de la formule logique ϕ.
- d'une système de transitions M, notée $|M|$, est donnée par $|M| = |S| + |S_0| + |R|$, i.e. le nombre d'états du système.
- d'un système discret $\mathcal{M}$, notée $|\mathcal{M}|$, est donnée par $|\mathcal{M}| = |V| + |\mathcal{S}_0| + |\mathcal{R}|$.

Lang. de modélisation/Sys. Discrets

- Propriété. Soit $\mathbb{M}$ la description d'un système dans l'un des langages de modélisation précédents (langage de Manna Pnueli, ou langage graphique, ou autre). $\mathbb{M}$ se traduit en un système discret $\mathcal{M}$ de taille polynomialement équivalente:

$$|\mathcal{M}| = O(|\mathbb{M}|^k).$$

  Et réciproquement.

- Exemples:
  - Peterson.
  - Compteur modulo 8.

Lang. de modélisation/Sys. Discrets

- Un système discret $\mathcal{M}$ correspond à un système de transitions M.
- Le système discret $\mathcal{M} = (V, \mathcal{S}_0, \mathcal{R})$ correspond au système de transitions $M = (S, S_0, R)$ avec
  - $S = D^V$. I.e. les états sont les valuations des variables.
  - $S_0 = \{s | s \models \mathcal{S}_0\}$. I.e. les états initiaux sont les valuations des variables qui satisfont $\mathcal{S}_0$.
  - $R = \{(s, s') | (s, s') \models \mathcal{R}\}$. I.e. les transitions sont les couples de valuations des variables qui satisfont $\mathcal{R}$.
- Mais un système discret de taille $|\mathcal{M}|$ peut correspondre à un système de transitions de taille $|M| = O(2^{|\mathcal{M}|})$.

n-processus en parallèle.

$\mathcal{M} = (V, \mathcal{S}_0, \mathcal{R})$	$M = (S, S_0, R)$
$V = \{x_1, \ldots, x_n\}$	$S = 2^n$
$D_{x_i} = \{0, 1\}$	
$\mathcal{S}_0 = \bigwedge_{i=1}^{n} x_i = 0$	$S_0 = \{s \mid s \models \mathcal{S}_0\}$
$\mathcal{R} = \bigvee_{i=1}^{n} x_i' = \neg x_i$	$R = \{(s, s') \mid (s, s') \models \mathcal{R}\}$
$\lvert\mathcal{M}\rvert = O(n^2)$	$\lvert M\rvert = O(2^n)$

Ce phénomène s'appelle l'explosion combinatoire.

- Suite de ce cours:
  - Étudier ce phénomène.
- En particulier, étudier formellement la difficulté de vérifier $\mathcal{M} \models \phi$ pour
  1. $\mathcal{M}$: système discret,
  2. ϕ: propriété d'invariance d'un prédicat d'état (i.e. ϕ de la forme $\forall\Box p$).

Définitions

- Soit $\mathcal{M} = (V, \mathcal{S}_0, \mathcal{R})$ un système discret, et $M = (S, S_0, R)$ le système de transitions correspondant.
- Un chemin (resp. initialisé) π de $\mathcal{M}$ est un chemin (resp. initialisé) de M.
- Un état $t \in S$ est atteignable s'il existe un chemin initialisé qui se termine en t.

Définitions (suite)

- Le sous-graphe atteignable d'un système de transitions $M = (S, S_0, R)$ est le système de transitions $M^R = (S^R, S_0, R^R)$,
  - dont les états S^R sont les états atteignables de M.
  - dont les transitions R^R sont les transitions atteignables. I.e. la restriction de R à S^R.
- Problème de l'atteignabilité:
  - Données:
    1. un système de transitions M,
    2. une région but $\sigma \subset S$.
  - Réponse:
    - Oui s'il existe une trajectoire initialisée qui atteint un état de σ, Non sinon.

Etats atteignables: exemple

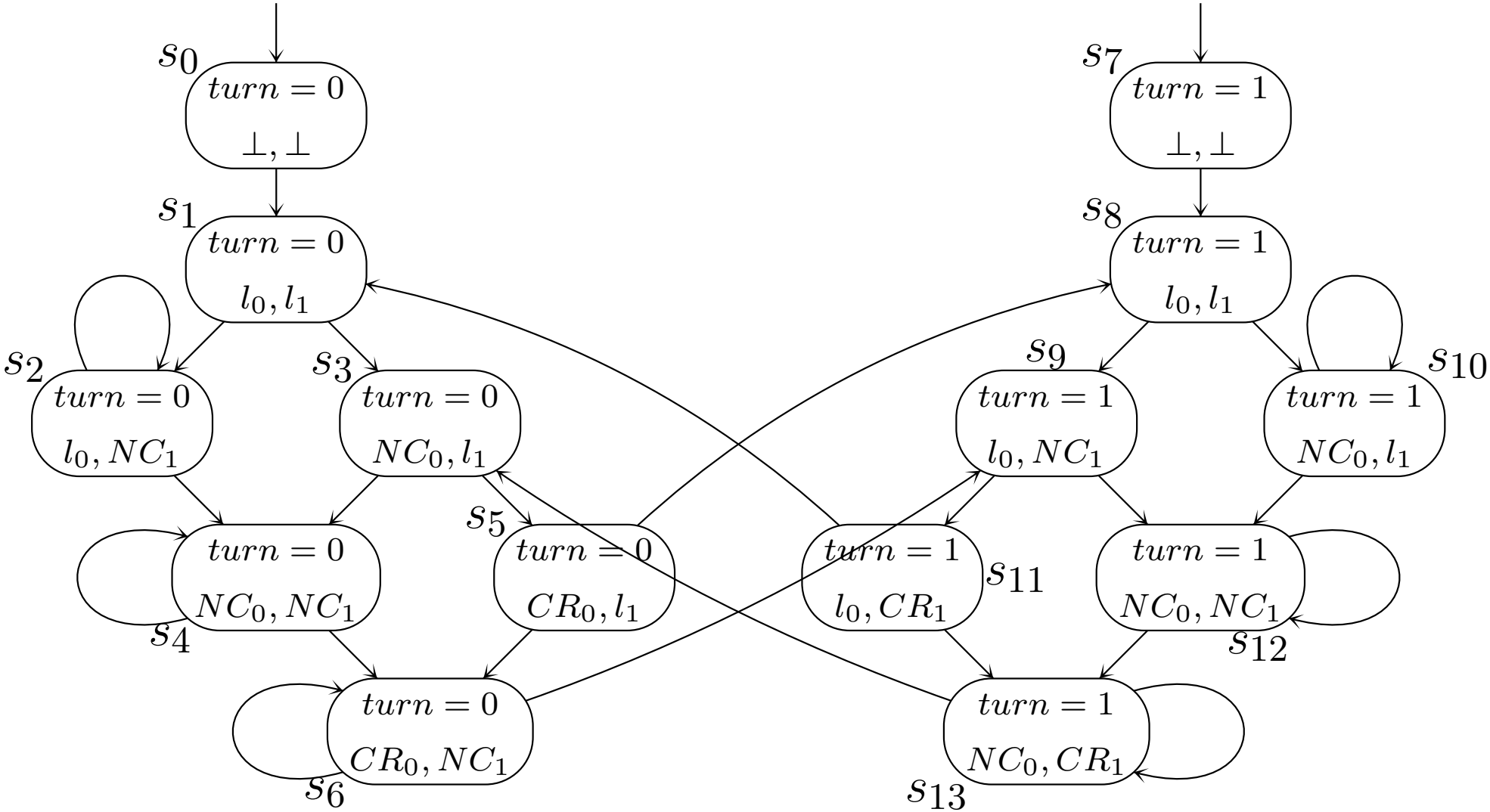

Figure 2: Exclusion mutuelle simple

Vérification d'invariant

- Un prédicat p sur les états définit la région $[\![p]\!] \subset S$ constituée des états s tels que $s \models p$.
- Problème de la vérification d'invariant:
  - Données:
    1. un système discret $\mathcal{M}$,
    2. un prédicat p.
  - Réponse:
    - Oui si $\mathcal{M} \models \forall\Box p$, Non sinon.

- Théorème: Le problème de vérification d'invariant $(\mathcal{M}, p)$ se ramène au problème d'atteignabilité $(M, [\![\neg p]\!])$.

  En effet, $\mathcal{M}$ vérifie $\forall\Box p$ ssi il n'existe pas de trajectoire initialisée qui atteint $[\![\neg p]\!]$.

- Conséquence: on peut vérifier une propriété d'invariance d'un système discret en résolvant un problème d'atteignabilité.

Problème de l'atteignabilité

Un problème d'atteignabilité peut se résoudre par un parcours de graphe:

- le problème de l'atteignabilité sur (M, σ) se ramène à tester si $S^R \cap \sigma \neq \emptyset$ où S^R est l'ensemble des états atteignables.
- S^R se calcule par un parcours de graphe (parcours en largeur, parcours en profondeur).

Parcours de graphe

- Schéma de l'algorithme:
  1. Données: $M = (S, S_0, R)$.
  2. Résultat: S^R région atteignable de M.
  3. Algorithme:

     $S^R = \emptyset$.

     $\tau = S_0$.

     tant que $\tau \neq \emptyset$

         choisir $s \in \tau$, et supprimer s de τ.

         si $s \notin S^R$

             $S^R := S^R \cup \{s\}$.

             $\tau := \tau \cup \{t | R(s,t)\}$.

- Implémentation:
  - τ pile $\rightsquigarrow$ parcours en profondeur.
  - τ file $\rightsquigarrow$ parcours en largeur.
- Complexité:
  - polynomiale: au plus $O(|S_0| + |S^R|)$ itérations.
  - $O(|S| + |R|)$.
  - ($NLOGSPACE$).

Vérification d'invariants

- Différence: on prend en entrée un système discret $\mathcal{M}$ et non pas un système de transitions M.
- Analyse précise de complexité:
  - Un système discret propositionnel est un système discret $\mathcal{M} = (V, \mathcal{S}_0, \mathcal{R})$ où
    1. toutes les variables $x \in V$ sont propositionnelles (i.e. de domaine $D_x = \{0, 1\}$).
    2. $\mathcal{S}_0$, $\mathcal{R}$ sont des expressions booléennes entre formules atomiques du type $\phi : x = 0 | x = 1 | true | false | \phi \wedge \phi | \phi \vee \phi | \neg\phi | \phi \leftrightarrow \phi | \phi \rightarrow \phi$.
    3. la propriété p d'invariance que l'on cherche à vérifier est aussi de ce type.

Vérification d'invariants (suite)

- Remarque: un système discret $\mathcal{M}$ propositionnel avec k variables correspond au pire cas à un système de transitions M avec 2^k états et 4^k transitions.
- Conséquence: la solution naïve qui réduit la vérification d'invariant au problème de l'atteignabilité nécessite un temps $O(4^k)$ et un espace $O(2^k)$ au pire cas.
- Question: peut-on faire mieux?

Borne inférieure

- Théorème: le problème de la vérification d'invariant pour les systèmes discrets propositionnels est PSPACE-complet.

Démonstration:

1. Étape 1: réduction à partir de la définition de PSPACE-complétude.

   Exercice. Principe: soit P un algorithme en espace polynomial. Sur une entrée de taille a, P utilise au plus un nombre polynomial $p(a)$ de cases du ruban d'une machine de Turing T implémentant P. Construire un système discret propositionnel $\mathcal{M}$ qui décrit l'évolution des $p(a)$ cases du ruban de la machine T. Soit ϕ la propriété "P n'est pas dans un état accepteur". Montrer que $\mathcal{M}$ vérifie $\forall\Box\phi$ ssi l'entrée est accepté par P. Conclure.

2. Étape 2: algorithme en espace polynomial.

   Voir plus loin.

Rappels sur les classes de complexité

- Classes définies en bornant l'espace ou le temps des machines de Turing.
- $NL \subset P \subset NP \subset PSPACE \subset EXPTIME$
  1. NL: Espace logarithmique.
  2. P: Temps polynomial.
  3. NP: Temps polynomial non-détérministe.
  4. $PSPACE$: Espace polynomial.
  5. $EXPTIME$: Temps exponentiel.
- Problème NP-complet canonique: satisfiabilité d'une formule propositionnelle.
- Problème PSPACE-complet canonique: satisfiabilité d'une formule propositionnelle quantifiée.

Un algorithme en espace polynomial.

- Algorithme
  - Données: système de transitions $M = (S, S_0, R)$, état s.
  - Résultat: Oui si s atteignable, non sinon.
  - Algorithme:
    1. pour tout état t

       si $t \in S_0$ alors

       si $Check(t, s, 2^n)$ retourner Vrai.

       retourner Faux.
    2. fonction $Check(t, u, i) : bool$
       - Données: états t, u, entier $1 \leq i \leq 2^n$.
       - Résultat: $\exists$ trajectoire de t vers u longueur $< i$?

Fonction $Check(t, u, i) : bool$

si $t = u$ return True.

si $R(t, u)$ return True.

si $i > 2$ alors

pour chaque état v

si $Check(t, v, \lfloor i/2 \rfloor + 1)$

si $Check(v, u, \lceil i/2 \rceil)$

retourner Vrai.

retourner Faux.

- Remarque: cet algorithme n'est JAMAIS utilisé en pratique car il fonctionne en temps *TOUJOURS* exponentiel en la taille du graphe atteignable. Alors que la technique précédente avait une complexité exponentielle seulement au pire cas.

Et pour des propriétés plus complexes

- Théorème: Le problème de la vérification de formules LTL pour les systèmes discrets propositionnels est PSPACE-complet.
- Théorème: Le problème de la vérification de formules CTL pour les systèmes discrets propositionnels est PSPACE-complet.
- Théorème: Le problème de la vérification de formules CTL* pour les systèmes discrets propositionnels est PSPACE-complet.

Remarques:

1. pour CTL,LTL,CTL* l'algorithme de vérification est linéaire en la taille du système de transitions mais pas en la taille du système discret.
2. Rappel: CTL temps $O(|\phi|(|S| + |R|))$, LTL,CTL* temps $O(2^{|\phi|}(|S| + |R|))$.
3. pour LTL et CTL*, on peut montrer que l'algorithme de vérification est aussi PSPACE-complet en la taille de la formule. Principe de la démonstration: montrer que la validité/satisfiabilité d'une formule LTL est un problème PSPACE-complet.

Conséquences

- Conséquence: l'explosion combinatoire est inéluctable.

  (en fait, sauf si $P = PSPACE$, qui implique $P = NP$, ce que beaucoup de gens conjecturent comme faux.)

- On a besoin de techniques de réduction de l'explosion combinatoire.
  1. Hachage
  2. Exploration à la volée
  3. Abstraction
  4. Réduction de graphes
  5. Réduction ordre partiel

Et pour des systèmes plus complexes

- Un système discret à compteur est $\mathcal{M} = (V, \mathcal{S}_0, \mathcal{R})$ avec
  1. $V = \{x_1, \ldots, x_d\}$ où chaque variable x_i a pour domaine $D_{x_i} = \mathbb{N}$.
  2. $\mathcal{S}_0, \mathcal{R}$ sont des expressions booléennes de formules atomiques du type $x = 0 | x > 0 | x' = 0 | x' = x + 1 | x' = x - 1$.
  3. La propriété p d'invariance à tester est du type $x = 0$.

- Théorème: Le problème de la vérification d'invariant pour les systèmes discrets à compteur est indécidable.

  Démonstration: toute machine de Turing se simule par une machine/système discrèt à deux compteurs. Cf par exemple "Introduction to Automata Theory Languages and Computation", Hopcroft/Ulmann, Addison-Wesley, 1979.

- Conséquence: Le problème de la vérification d'invariant pour les systèmes discrets à nombre d'états non-bornés est indécidable.

  (Autre démonstration: le problème de l'arrêt d'une machine de Turing peut se voir comme un problème de vérification d'invariant particulier.)