Storm

A probabilistic model checker

Florent Delgrange

UMONS

Faculté des Sciences

Mab2 Science Informatique

Plan

Modèles

- Chaines de Markov à temps discret (CM)
- Chaines de Markov à temps continu (CMC)
- Processus décisionnels de Markov (PDM)
- Automates de Markov (AM)

Chaines de Markov à temps continu

Definition (Chaîne de Markov à temps continu)

Une chaîne de Markov $\mathcal{M}$ à temps continu est définie par

- un ensemble d'états S
- une fonction de transition initiale $\mathcal{Q} : S \times S \to [0, 1] \cap \mathbb{Q}$, définissant une **distribution de probabilité intiale** sur les états S
- une fonction de taux d'intensité $\lambda : S \to \mathbb{R}$
- une fonction génératrice $\mathcal{A} : S \times S \to [0, 1]$

Chaînes de Markov à temps continu

Soient $s, s' \in S$, deux états de $\mathcal{M}$ et T_s, le temps passé passé par le système dans l'état s

- $\mathbb{P}(T_s \geq t) = e^{-\lambda(s)t}$
- La distribution **exponentielle négative** est la **seule à respecter la propriété de Markov** (états sans mémoire) !
- $\mathbb{E}(T_s) = \frac{1}{\lambda(s)}$
- $\mathcal{A}(s, s') = \begin{cases} -\lambda(s) & \text{si } s = s' \\ \lambda(s) \cdot \mathcal{Q}(s, s') & \text{sinon} \end{cases}$

  Remarque : $\mathcal{Q}(s, s') = 0$ en temps continu

Chaînes de Markov à temps continu

Supposons à présent que l'ensemble des états S de la chaîne est **fini**. Alors, la fonction $\mathcal{A}$ peut être caractérisée par une matrice génératrice A, telle que

$$\mathcal{A}(s_i, s_j) = A_{i,j}$$

Dès lors, soit la fonction $P : \mathbb{R}^+ \to [0, 1]^{|S| \times |S|}$, la fonction générant la matrice caractérisant la fonction de transition $\Delta : S \times S \times \mathbb{R}^+ \times [0, 1]$ selon un temps $t \in \mathbb{R}^+$,

$$P(t) = \begin{cases} \mathbb{1} & \text{si } t = 0 \\ e^{At} & \text{sinon} \end{cases}$$

$$\Leftrightarrow \Delta(s_i, s_j, t) = P(t)_{i,j}$$

Plan

Format d'entrées

- PRISM
- JANI
- Réseaux de Petri stochastiques généralisés (GSPN)
- Dynamic Fault Trees (DFTs)
- pGCL
- Explicite

PRISM

Syntaxe

Soit $\mathcal{M} = (S, A, \Delta, w)$, un PDM.

Énumération et initialisation des états

Soit $X \subseteq S$, avec $|X| = n \in \mathbb{N}$

$$x : [0 \ldots n-1] \; init \; 0$$

avec x_0, un état initial de $\mathcal{M}$

PRISM

Syntaxe

Soit $\mathcal{M} = (S, A, \Delta, w)$, un PDM.

Transitions

Soit $s_i \in S$, le $i^{\text{ème}}$ état de S. Pour toute action $\alpha \in A(s)$,

$$[\alpha] \quad s = i \rightarrow \delta_1 : s' = j_1 + \cdots + \delta_m : s' = j_m$$

Avec

- $s_{j_1}, \ldots s_{j_m}$, les α-successeurs de s_i
- $\delta_k = \Delta(s_i, \alpha, s_{j_k})$

PRISM

Syntaxe

Soit $\mathcal{M} = (S, A, \Delta, w)$, un PDM.

Rewards

Soient une action $\alpha \in A$ et Φ, une formule d'états de S,

$$[\alpha] \quad \Phi : r$$

r correspond à la *récompense* du choix de l'action α menant aux états satisfaisant Φ.

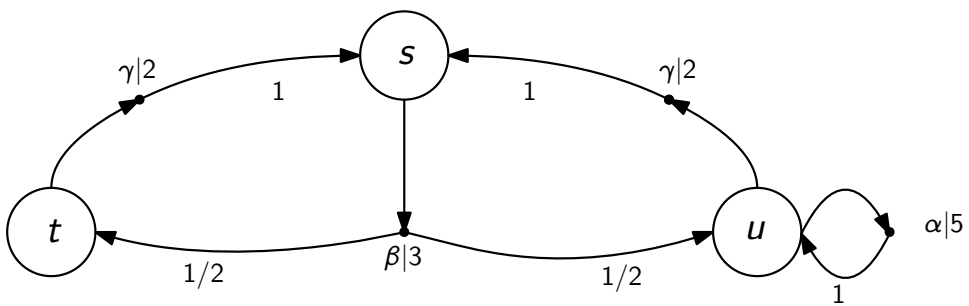
Note : Le poids de l'action α, i.e., $w(\alpha)$, est exprimé par

$$[\alpha] \quad true : r$$

PRISM

Exemple

```
1   mdp
2
3   module classic
4
5   s: [0..2] init 0;
6
7   [beta] s=0 -> 0.5 : (s'=1) + 0.5 : (s'=2);
8   [gamma] s=1 -> 1 : (s'=0);
9   [alpha] s=2 -> 1 : (s'=2);
10  [gamma] s=2 -> 1 : (s'=0);
11
12  endmodule
13
14  label "T" = s=1;
15
16  rewards "weights"
17    [alpha] true : 5;
18    [beta] true : 3;
19    [gamma] true : 2;
20  endrewards
```



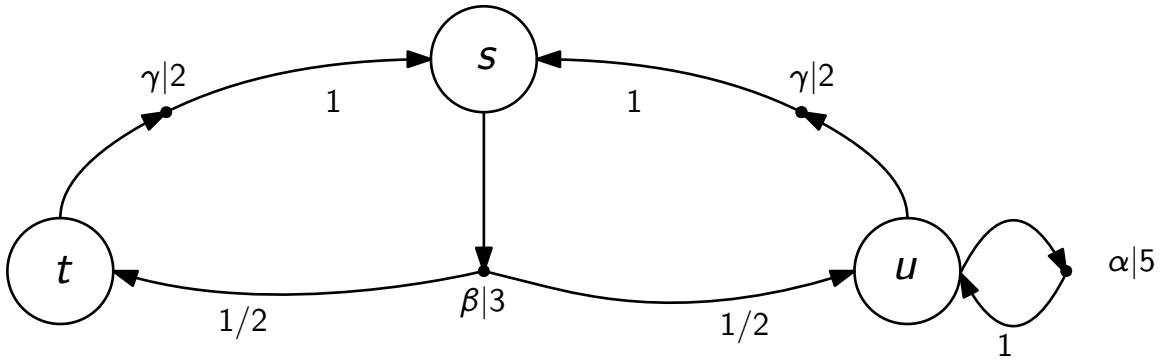
Explicite

Soit $\mathcal{M} = (S, A, \Delta, w)$, un PDM. $\mathcal{M}$ est représenté par

- P, une matrice de transition de 4 colonnes telle que chaque ligne exprime la probabilité d'une transition.
  Exemple : $P_{i.} = 0\ 0\ 1\ 0.3$ signifie que l'état $s_0 \in S$ rejoint l'état s_1 lorsque l'actions α_0 est choisie avec une probabilité $\frac{3}{10}$
- R, une matrice de récompenses (*rewards transition matrix*) de taille 4 telle que chaque ligne exprime la récompense d'une transition.
  Exemple : $R_{i.} = 0\ 0\ 1\ 5$ signifie que la transition de l'état $s_0 \in S$ en l'état s_1 lorsque l'actions α_0 est choisie induit une récompense de 5.

Explicite

Exemple



$$P = \begin{pmatrix} 0 & 1 & 1 & \frac{1}{2} \\ 0 & 1 & 2 & \frac{1}{2} \\ 1 & 2 & 0 & 1 \\ 2 & 0 & 2 & 1 \\ 2 & 2 & 0 & 1 \end{pmatrix} \qquad R = \begin{pmatrix} 0 & 1 & 1 & 3 \\ 0 & 1 & 2 & 3 \\ 1 & 2 & 0 & 2 \\ 2 & 0 & 2 & 5 \\ 2 & 2 & 0 & 2 \end{pmatrix}$$

Plan

Propriétés

États

PrCTL (Prism-CTL)

Soient Φ, Ψ, des formules d'état,

- $\Phi\, U\, \Psi$ Φ until Ψ
- $F\Phi \equiv true\, U\, \Phi$
- $\Phi U\{\mathrm{op}\}t\, \Psi$ (après t étapes en temps discret et t unités de temps en continu)
- $F\{\mathrm{op}\}\, \Phi \equiv true\, U\{\mathrm{op}\}\Phi$
- $G\Phi$ exprime les chemins pour lesquel Φ est vrai à chaque étape

Propriétés

Chemins

PrCTL (Prism-CTL)

Soient Φ, Ψ, des formules d'état,

- $P\{\text{op}\}x[\Phi]$ probabilité des chemins satisfaisant Φ, avec x un seuil de probabilité
- $LRA\{\text{op}\}x[\Phi]$ propabilité des chemins satisfaisant Φ lorsque le système est en **état stationnaire**, avec x un seuil de probabilité
- $R\{\text{op}\}c[\Phi]$ coût attendu pour atteindre les chemins satisfaisant Φ, avec c, un seuil de coût

Pour les systèmes non-déterministes, min ou max doit être spécifié avant l'opérateur (e.g., *Pmax, Rmin*, etc.)

Propriétés

PrCTL

Exemples de propriétés :

- $P <= x [F\, T]$ $\equiv$ $\mathcal{P}_{\leq x} \Diamond T$, i.e., $FT = true\, UT$
- $P = x [F <= t\, T]$ $\equiv$ $\mathcal{P}_{=x} (\Diamond^{\leq t} T)$, i.e., T est atteint après t étapes (ou t unités de temps lorsque ce dernier est continu) dans une CM
- $Pmax = x [F\, T]$ $\equiv$ $\mathcal{P}^{\max}{}_{=x} \Diamond T$ (choix non-déterministes)
- $LRA <= x [\Psi]$, i.e., la probabilité que le système se situe en un des états qui satisfont la formule d'états Ψ lorsque le système est en *état stationnaire* (*long run average probability*) est inférieure à x
- $Rmin <= x [F\, T]$, i.e., l'espérance maximale pour atteindre un état de T est inférieure à x (choix non-déterministes)
- etc.

Propriétés

PrCTL

Exemples de requêtes :

- $P =? [F <= t\, T]$
- $Pmax =? [F\, T]$ (choix non-déterministes)
- $LRA <=? [\Psi]$
- $Rmin <=? [F\, T]$ (choix non-déterministes)
- etc.

Plan

Engines

- **Sparse** : représentation par matrices creuses $\Longrightarrow$ opérations rapides pour les modèles de taille modérée (le plus performant en moyenne)
- **Hybrid** : représentations par des *MTBDDs*, mais emploie parfois des matrices creuses si les opérations sont jugées plus appropriées avec ce format
- **DD** : représentation par *MTBDDs*
- **Abstraction-Refinement** : rend les modèles (pas forcément finis) de Markov à temps discret en jeux stochastiques (finis) et améliore l'abstraction si nécessaire.
- **Exploration** : représentation par matrice creuse. Ce modèle explore l'espace d'état avec des méthodes de machine learning.

Sparse

- **En général**, les graphes sous-jacent des modèles de Markov sont rarement complets (ou presque)

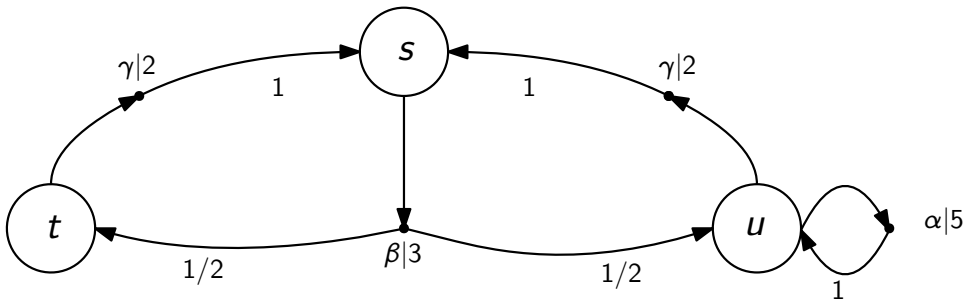
$\Longrightarrow$ Les matrices de transition représentant de tels modèles sont **creuses**

- des méthodes d'analyse numérique permettent des implémentations efficaces
  - pour stocker de telles matrices en mémoire
  - pour effectuer rapidement les opérations matricielles

Sparse

Représentation

Exemple de représentation (avec `stormpy`)




```
>> model.transition_matrix()

        0   1   2
   ---- group 0/2 ----
0   (   0   0.5 0.5     )   0
   ---- group 1/2 ----
1   (   1   0   0       )   1
   ---- group 2/2 ----
2   (   1   0   0       )   2
3   (   0   0   1       )   3
        0   1   2
```

Hybrid et DD

- *MTBDD* : Multi Terminal Binary Decision Diagram
- BDDs permettant des valeurs numériques comme valeur pour les feuilles
- DAG pour représenter de façon plus compacte les BDDs
- efficace pour diverses opérations matricielles

Arithmétique exacte

- Les méthodes numériques appliquées aux vérificateurs de modèles probabilistes (probabilistic model checkers) sont sujets à des problèmes numériques
- **STORM** permet d'activer le mode d'arithmétique exacte permettant d'obtenir des résultats précis

Arithmétique exacte

Exemple

- `stormpy 1.1.1` (désactivée par défaut)

```
formula = 'Rmin=? [F "T"]'
properties =  \
  stormpy.parse_properties_for_prism_program(
    formula, prism_program
  )
model = stormpy.build_model(prism_program, properties)
result = stormpy.model_checking(model, properties[0])
assert result.result_for_all_states
for s, x in enumerate(result.get_values()):
    print("v[{}]={}".format(s, x))
```

```
v[0]=7.999990463256836
v[1]=0.0
v[2]=9.999984741210938
```

Arithmétique exacte

Exemple

- `stormpy 1.2.0` (activée par défaut)

```
formula = 'Rmin=? [F "T"]'
properties =  \
  stormpy.parse_properties_for_prism_program(
    formula, prism_program
  )
model = stormpy.build_model(prism_program, properties)
result = stormpy.model_checking(model, properties[0])
assert result.result_for_all_states
for s, x in enumerate(result.get_values()):
    print("v[{}]={}".format(s, x))
```

```
v[0]=8.0
v[1]=0.0
v[2]=10.0
```

Solveurs

- **Équations de Bellman** utilisant les **matrices creuses** et les **MTBDDs**
- **MILP** (Mixed Integer Linear Programming)
- **Jeux stochastiques**

  Note : STORM ne supporte pas les jeux stochastiques en modèle d'entrée ; mais le *abstraction refinement engine* utilise ces représentations

Solveurs

- **SMT** (Satisfiability Modulo Theories)
  - problème de décision pour les formules de logique du premier ordre avec égalité et sans quantificateurs
  - But : déterminer si une telle formule est satisfiable par rapport à une théorie sous-jacente
  - Exemple :

$$(x < 0 \vee x > 1) \wedge (x = y + 5) \wedge (y > 0)$$

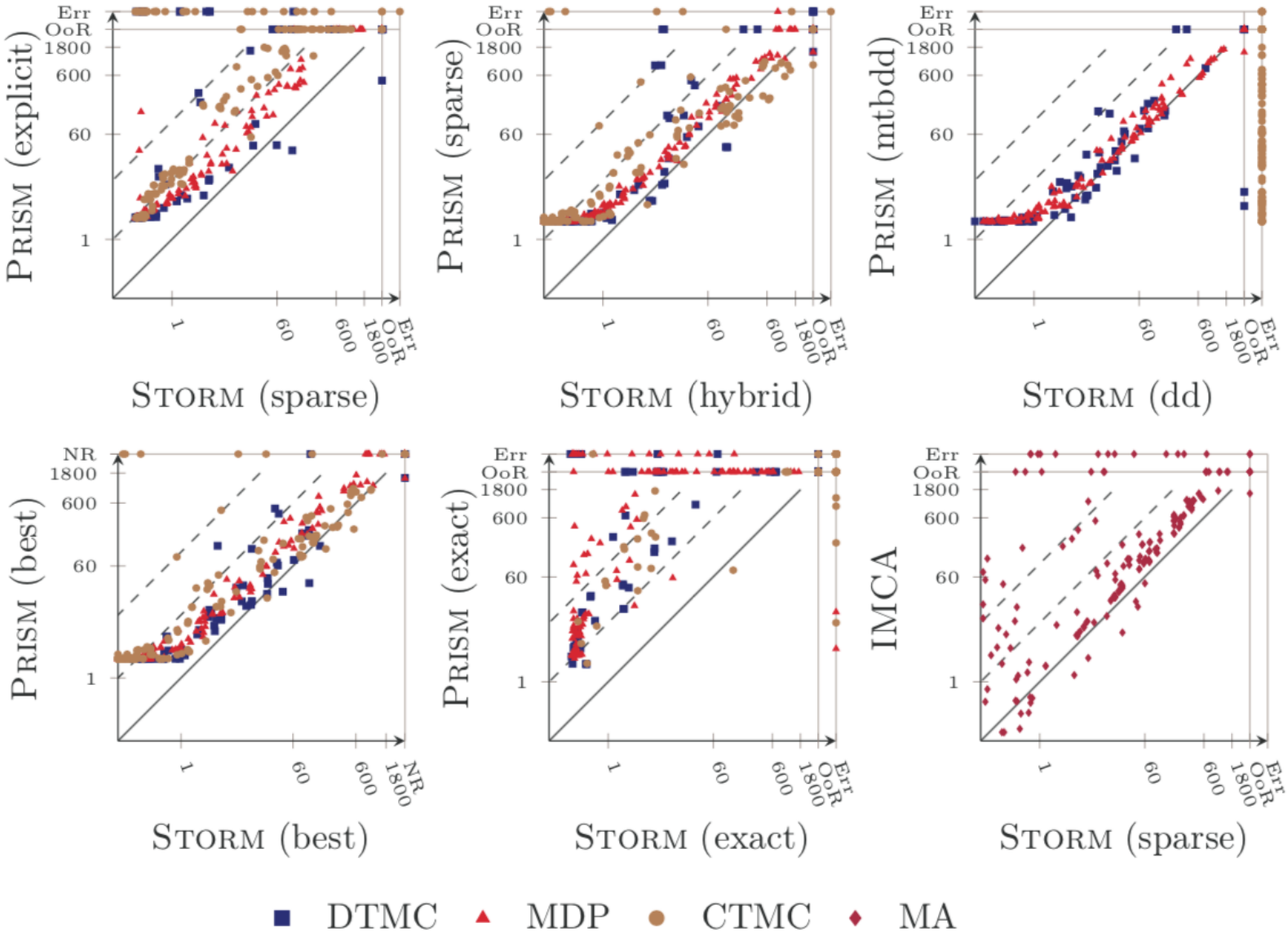
    satisfiable par rapport à la théorie des nombres réels ?
  - En pratique : employé pour minimiser la taille des modèles de Markov par *bisimulation* (curse of dimensionality).

Storm 1.2.0

- Nouvelle release (8 décembre 2017)
- Support pour les “reward bounded reachability properties” en multi-objectif
- Model checking par programmation linéaire pour les PDM
- Performances augmentées
- etc.

Plan

Résultats


PRISM (explicit)
STORM (sparse)
PRISM (sparse)
STORM (hybrid)
PRISM (mtbdd)
STORM (dd)
PRISM (best)
STORM (best)
PRISM (exact)
STORM (exact)
IMCA
STORM (sparse)
Err
OoR
NR
1800
600
60
1
DTMC
MDP
CTMC
MA