Synthèse dans les Jeux Quantitatifs Multi-Critères

Mickaël Randour
Promoteurs: Véronique Bruyère & Jean-François Raskin

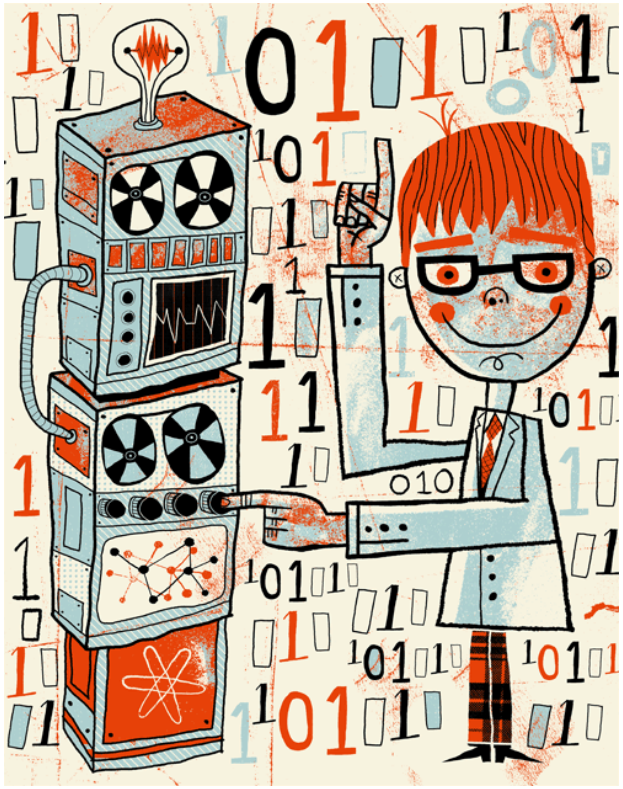
Mons - 24.04.2014

Défense publique de thèse

UMONS
Université de Mons

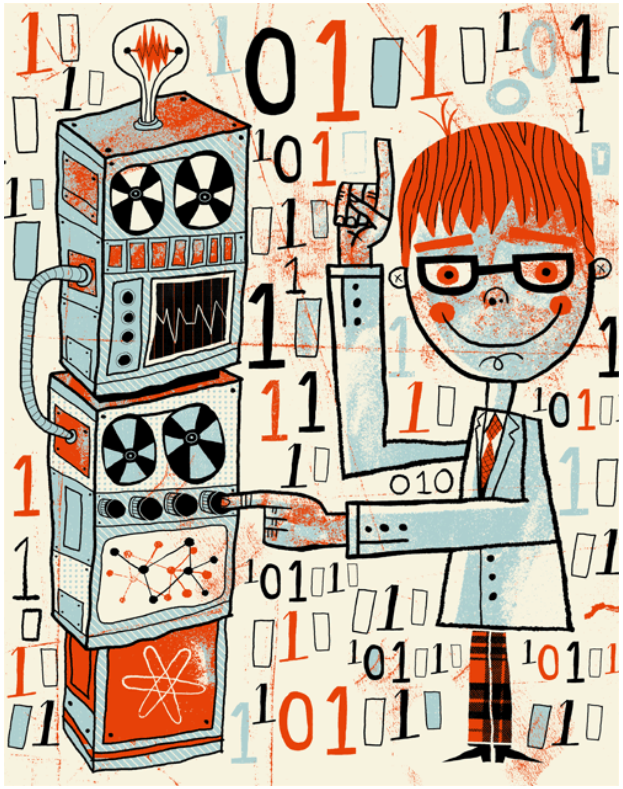
ULB UNIVERSITÉ LIBRE DE BRUXELLES

Informatique ? Oui, mais. . .

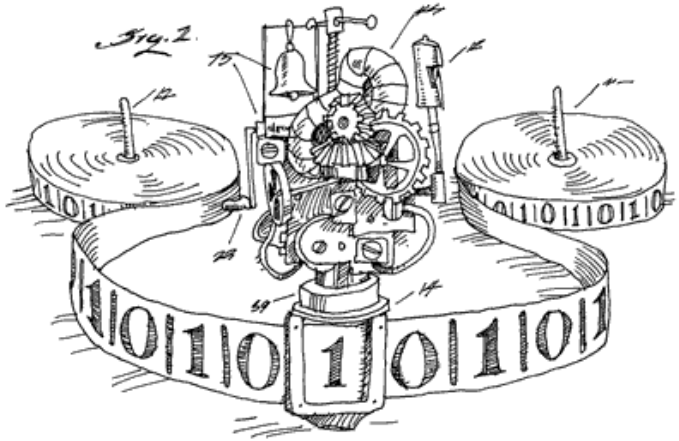


Je suis un informaticien.

Informatique ? Oui, mais. . .



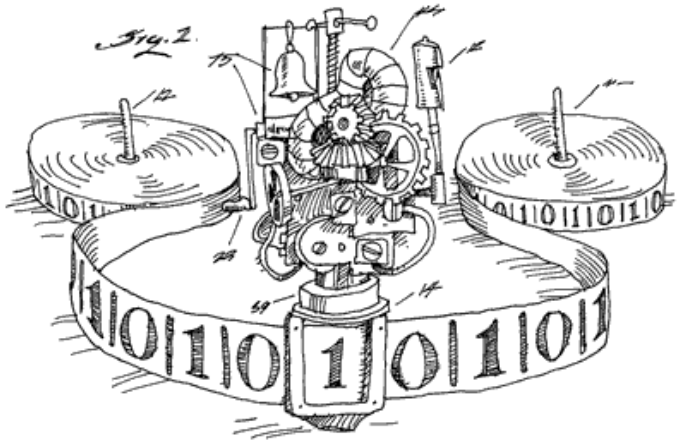
Je suis un informaticien.



Mais voici le type de machines que je manipule. Focus sur l'informatique **théorique**.

Informatique ? Oui, mais. . .

Alan Turing (1912-1954)

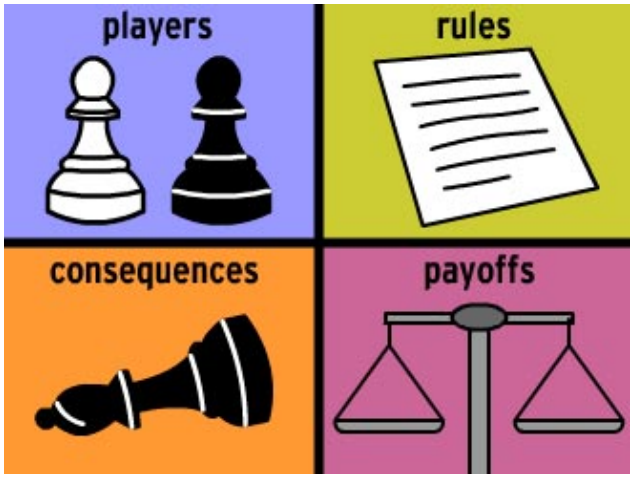


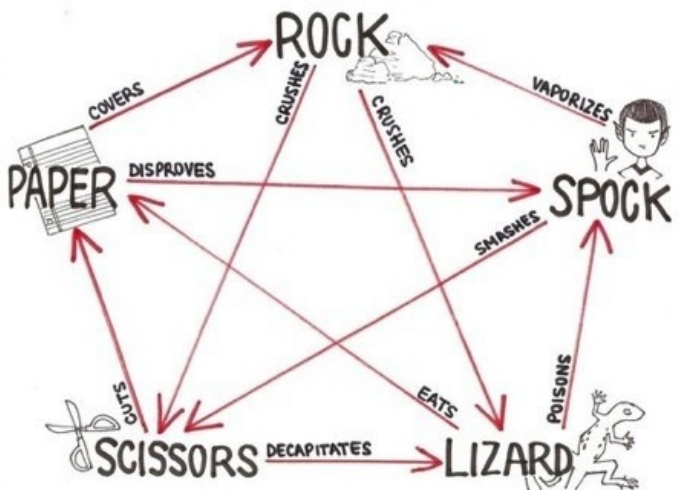
Mais voici le type de machines que je manipule. Focus sur l'informatique **théorique**.

Machine de Turing : modèle abstrait (mathématique) d'ordinateur.

Des jeux comme modèles ?

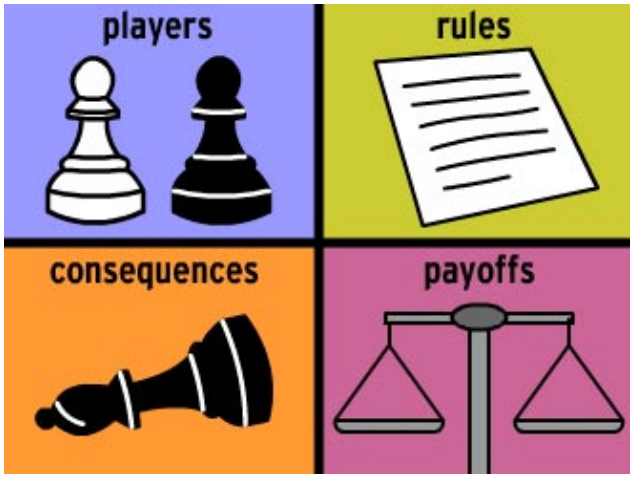
Mes outils sont des **jeux** [vNM44, OR94].

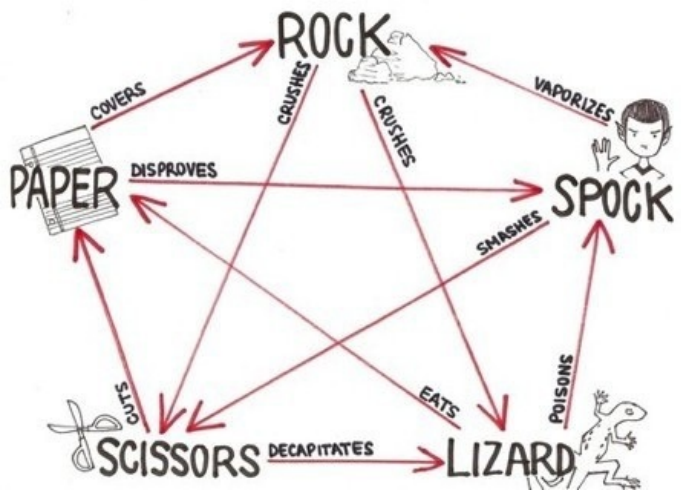




Des jeux comme modèles ?

Mes outils sont des **jeux** [vNM44, OR94].





Discussion de haut niveau. Je vais vous présenter

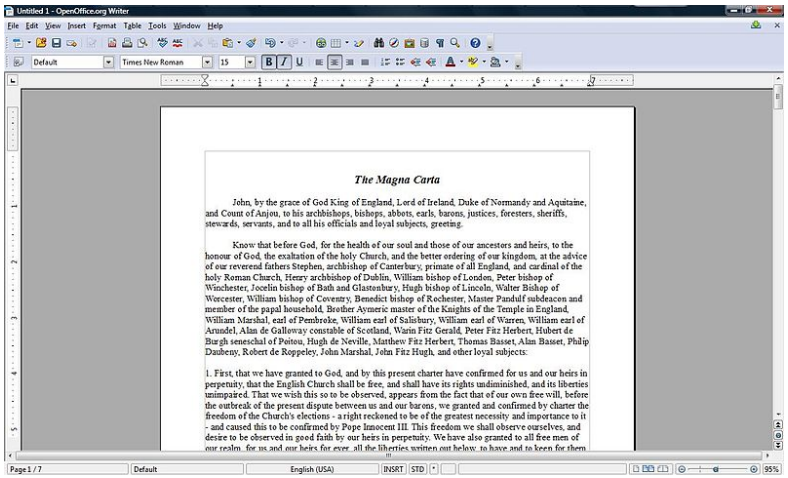
- à quoi ils *servent*,
- à quoi ils *ressemblent*,
- la nature des *questions* que nous avons *résolues*.

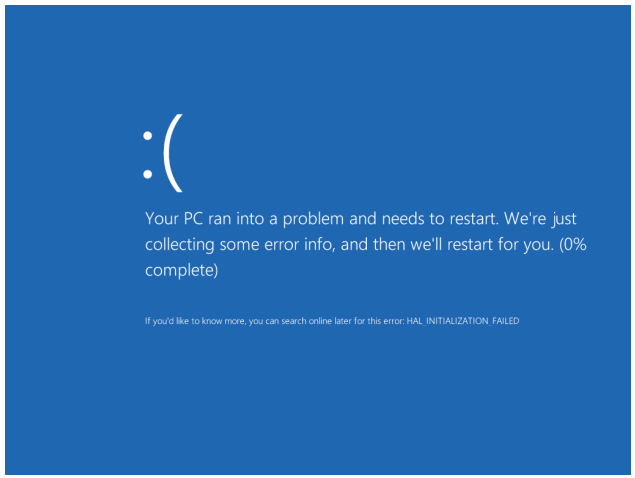
Système (informatique) réactif

- Interaction continue avec l'**environnement**, besoin de *réagir* aux événements.
- Systèmes très grands et complexes $\rightsquigarrow$ sujets aux bugs et erreurs.

Système (informatique) réactif

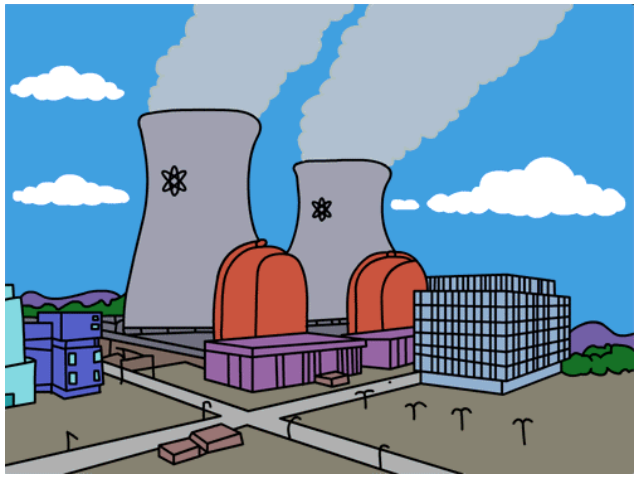
- Interaction continue avec l'**environnement**, besoin de *réagir* aux événements.
- Systèmes très grands et complexes $\rightsquigarrow$ sujets aux bugs et erreurs.
  - ▷ **Tester** pour détecter et corriger des défectuosités.
  - ▷ En cas de bug résiduel, on peut toujours développer un *patch* par la suite...





Systèmes critiques

- Certains systèmes **ne** tolèrent **pas** les bugs.
  - ▷ Tester ne suffit pas !



- Petits bugs, conséquences désastreuses !
  - ▷ Therac-25 radiothérapie : plusieurs morts.
  - ▷ Pentium II : $\sim$ 500 million \$.
  - ▷ Explosion d'Ariane 5 (overflow).
  - ▷ Perte de Mars Climate Orbiter (système impérial vs. métrique).

Preuves formelles de fiabilité

- Besoin de preuves mathématiques qu'un système aura un *comportement correct*, indépendamment du comportement de son environnement.

Preuves formelles de fiabilité

- Besoin de preuves mathématiques qu'un système aura un *comportement correct*, indépendamment du comportement de son environnement.
- **Spécification** : définit les comportements acceptables ou non du système.

Preuves formelles de fiabilité

- Besoin de preuves mathématiques qu'un système aura un *comportement correct*, indépendamment du comportement de son environnement.
- **Spécification** : définit les comportements acceptables ou non du système.
- Les systèmes complets sont trop complexes : besoin de **modèles** abstraits pour travailler.

Preuves formelles de fiabilité

- Besoin de preuves mathématiques qu'un système aura un *comportement correct*, indépendamment du comportement de son environnement.
- **Spécification** : définit les comportements acceptables ou non du système.
- Les systèmes complets sont trop complexes : besoin de **modèles** abstraits pour travailler.
- Deux approches : **vérification** et **synthèse**.

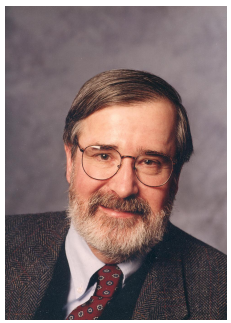
Vérification

Vérification : vérifier qu'un (modèle de) système existant satisfait une spécification donnée, processus *a posteriori*.

Vérification

Vérification : vérifier qu'un (modèle de) système existant satisfait une spécification donnée, processus *a posteriori*.

- ▷ **Model checking** $\sim$ 1980 [CE81, VW86, CGP00, BK08].


E. Clarke


A. Emerson


J. Sifakis

Prix Turing 2007


M. Vardi

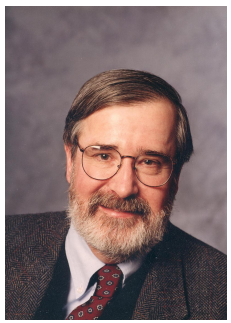

P. Wolper

Prix Gödel 2000

Vérification

Vérification : vérifier qu'un (modèle de) système existant satisfait une spécification donnée, processus *a posteriori*.

- ▷ **Model checking** $\sim$ 1980 [CE81, VW86, CGP00, BK08].


E. Clarke


A. Emerson


J. Sifakis

Prix Turing 2007


M. Vardi


P. Wolper

Prix Gödel 2000

- ▷ Outils puissants, établis dans l'industrie (p.ex. IBM, Airbus).
- ▷ Liens forts avec la *logique mathématique* et la *théorie des automates*. Voir travaux de Büchi, Rabin, etc [Büc60, Rab69].

Synthèse

Synthèse : construire de manière automatisée un système correct *à partir* de la spécification, processus *a priori*.

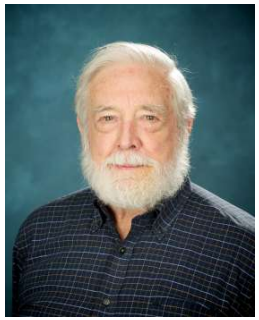
Synthèse

Synthèse : construire de manière automatisée un système correct *à partir* de la spécification, processus *a priori*.

- ▷ Plus ambitieux, mais **plus difficile** [Chu57, RW87, PR89].


A. Church


P. Ramadge


W. Wonham


A. Pnueli

Prix Turing 1996


R. Rosner

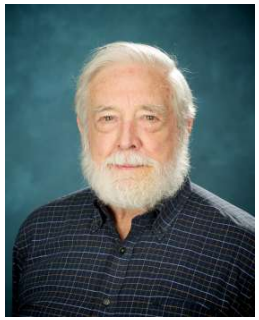
Synthèse

Synthèse : construire de manière automatisée un système correct *à partir* de la spécification, processus *a priori*.

- Plus ambitieux, mais **plus difficile** [Chu57, RW87, PR89].


A. Church


P. Ramadge


W. Wonham


A. Pnueli
Prix Turing 1996


R. Rosner

- Dans cette thèse, nous étudions la synthèse de **contrôleurs** fiables à l'aide de la *théorie des jeux*.

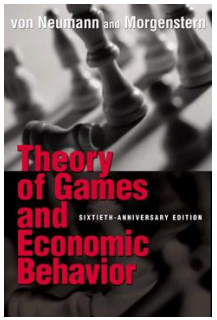
Théorie des jeux

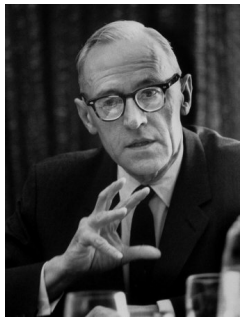
- Interactions entre entités ou systèmes vues comme des jeux entre plusieurs joueurs rationnels.
- Applications en informatique, économie, biologie, politique. . .

Théorie des jeux

- Interactions entre entités ou systèmes vues comme des jeux entre plusieurs joueurs rationnels.
- Applications en informatique, économie, biologie, politique...

J. von Neumann

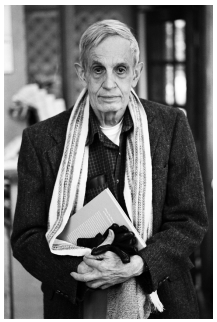




O. Morgenstern

$\neq$

E. Zermelo



$\sim$

J. Nash

Assurer la fiabilité via des jeux

- Modéliser les interactions entre le système et son environnement dans un jeu.
- Le système essaie de fonctionner correctement.
- L'environnement essaie de l'en empêcher.

Assurer la fiabilité via des jeux

- Modéliser les interactions entre le système et son environnement dans un jeu.
- Le système essaie de fonctionner correctement.
- L'environnement essaie de l'en empêcher.

▷ Si on trouve une stratégie qui gagne à tous les coups pour le système, alors on sait comment contrôler le système pour qu'il soit toujours fiable, quoiqu'il arrive.

Assurer la fiabilité via des jeux

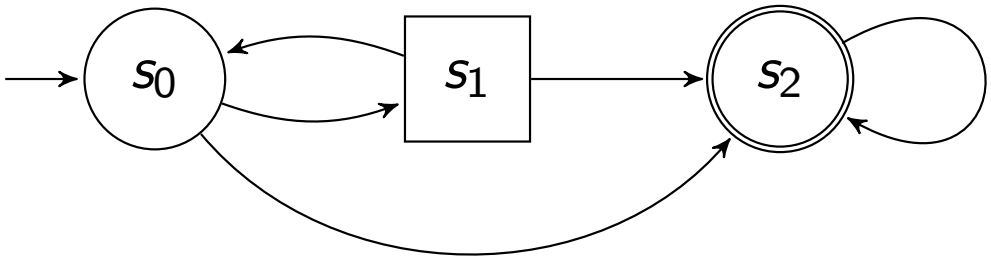
- Modéliser les interactions entre le système et son environnement dans un jeu.
- Le système essaie de fonctionner correctement.
- L'environnement essaie de l'en empêcher.

▷ Si on trouve une stratégie qui gagne à tous les coups pour le système, alors on sait comment contrôler le système pour qu'il soit toujours fiable, quoiqu'il arrive.

▷ Cadre mathématique pour prouver formellement qu'un système est correct, et pour construire des systèmes corrects de manière automatisée.

Jeux sur graphes

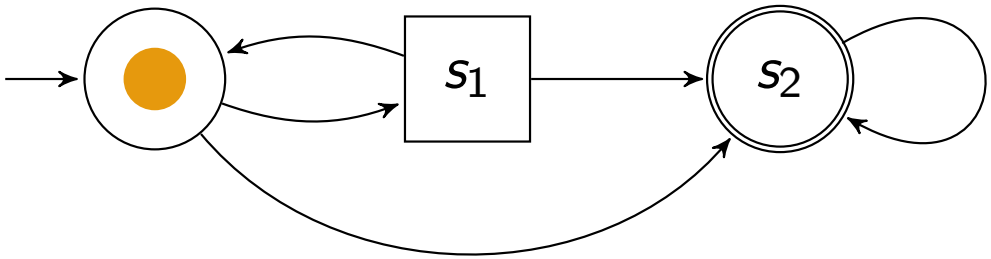
- Modèlisent l'interaction entre deux joueurs : le système (○) et son adversaire, l'environnement (□).



▷ *Etats* et *transitions*.

Jeux sur graphes

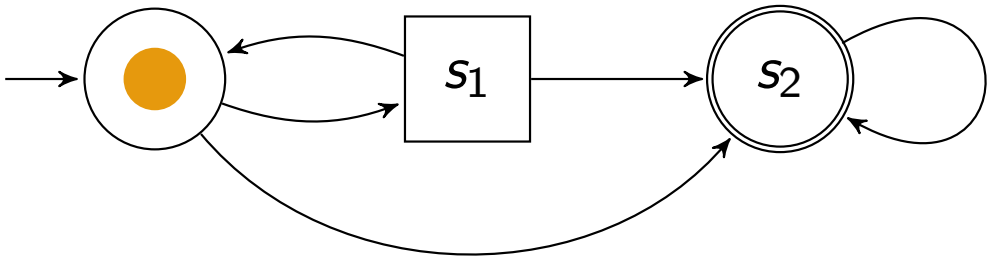
- Modèlisent l'interaction entre deux joueurs : le système (○) et son adversaire, l'environnement (□).



▷ Une *partie* démarre dans un état initial : imaginons un jeton désignant l'état courant.

Jeux sur graphes

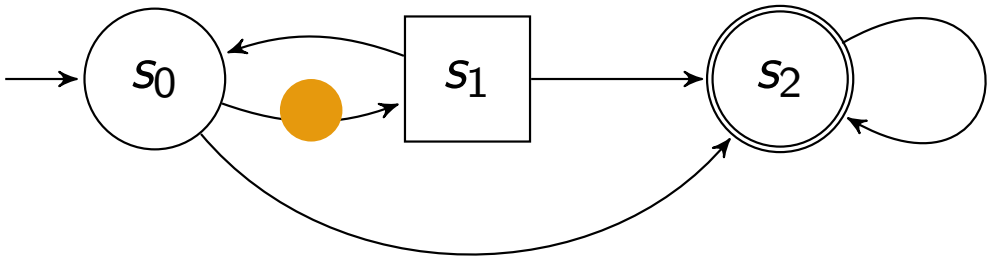
- Modèlisent l'interaction entre deux joueurs : le système (○) et son adversaire, l'environnement (□).



- ▷ Les joueurs jouent l'un après l'autre : le propriétaire d'un état décide où va le jeton.
- ▷ Les joueurs suivent des *stratégies* : des fonctions des histoires vers des choix. Potentiellement complexes ! Peuvent utiliser de la mémoire et des choix aléatoires.

Jeux sur graphes

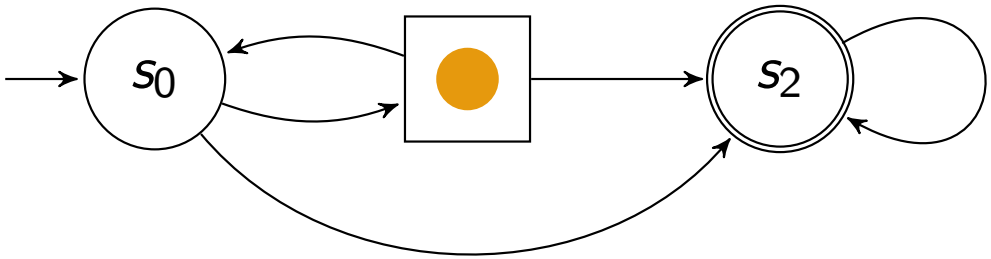
- Modèlisent l'interaction entre deux joueurs : le système (○) et son adversaire, l'environnement (□).



- ▷ Les joueurs jouent l'un après l'autre : le propriétaire d'un état décide où va le jeton.
- ▷ Les joueurs suivent des *stratégies* : des fonctions des histoires vers des choix. Potentiellement complexes ! Peuvent utiliser de la mémoire et des choix aléatoires.

Jeux sur graphes

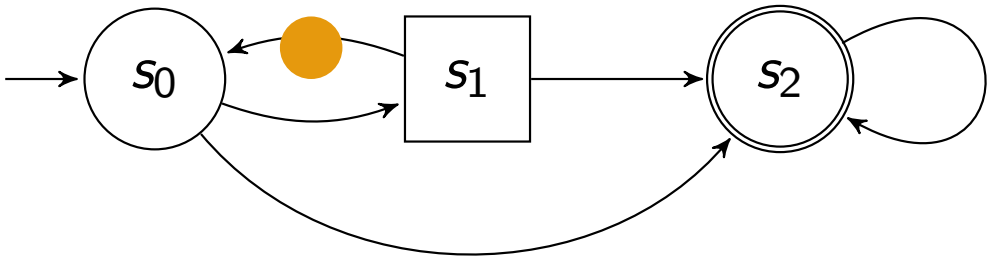
- Modèlisent l'interaction entre deux joueurs : le système (○) et son adversaire, l'environnement (□).



- ▷ Les joueurs jouent l'un après l'autre : le propriétaire d'un état décide où va le jeton.
- ▷ Les joueurs suivent des *stratégies* : des fonctions des histoires vers des choix. Potentiellement complexes ! Peuvent utiliser de la mémoire et des choix aléatoires.

Jeux sur graphes

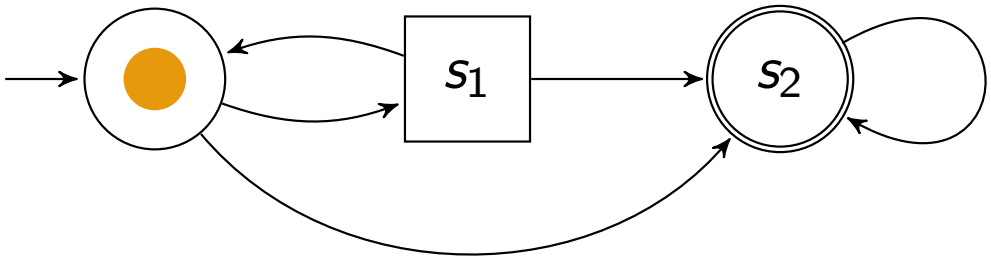
- Modèlisent l'interaction entre deux joueurs : le système (○) et son adversaire, l'environnement (□).



- ▷ Les joueurs jouent l'un après l'autre : le propriétaire d'un état décide où va le jeton.
- ▷ Les joueurs suivent des *stratégies* : des fonctions des histoires vers des choix. Potentiellement complexes ! Peuvent utiliser de la mémoire et des choix aléatoires.

Jeux sur graphes

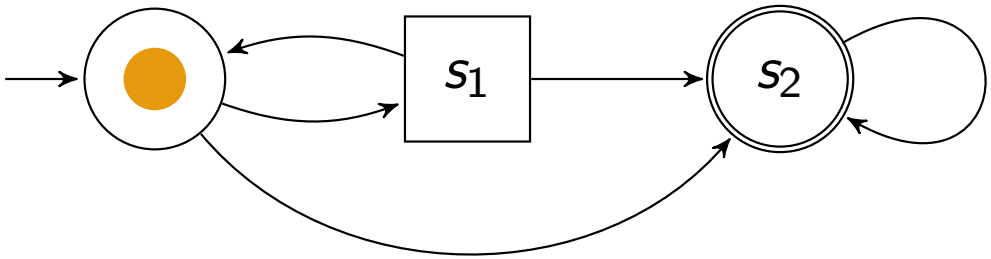
- Modèlisent l'interaction entre deux joueurs : le système (○) et son adversaire, l'environnement (□).



- ▷ Les joueurs jouent l'un après l'autre : le propriétaire d'un état décide où va le jeton.
- ▷ Les joueurs suivent des *stratégies* : des fonctions des histoires vers des choix. Potentiellement complexes ! Peuvent utiliser de la mémoire et des choix aléatoires.

Jeux sur graphes

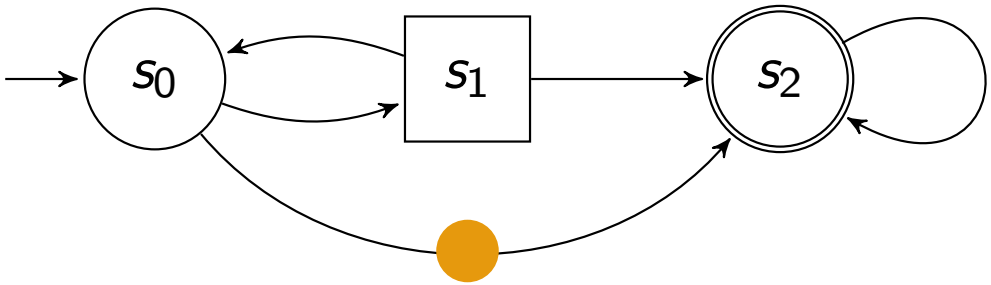
- Modèlisent l'interaction entre deux joueurs : le système (○) et son adversaire, l'environnement (□).



- ▷ Les parties sont infinies. Elles sont déclarées *gagnantes* pour le système si elles *satisfont la spécification* (= comportement acceptable). Sinon, l'environnement gagne.
- ▷ P.ex. s_2 doit être visité infiniment souvent.

Jeux sur graphes

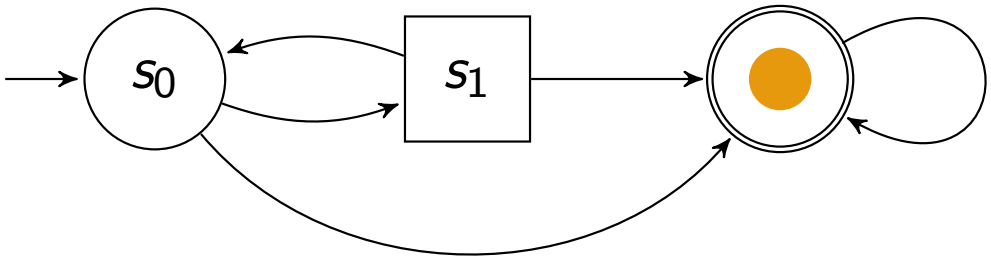
- Modèlisent l'interaction entre deux joueurs : le système (○) et son adversaire, l'environnement (□).



- ▷ Les parties sont infinies. Elles sont déclarées *gagnantes* pour le système si elles *satisfont la spécification* (= comportement acceptable). Sinon, l'environnement gagne.
- ▷ P.ex. s_2 doit être visité infiniment souvent.

Jeux sur graphes

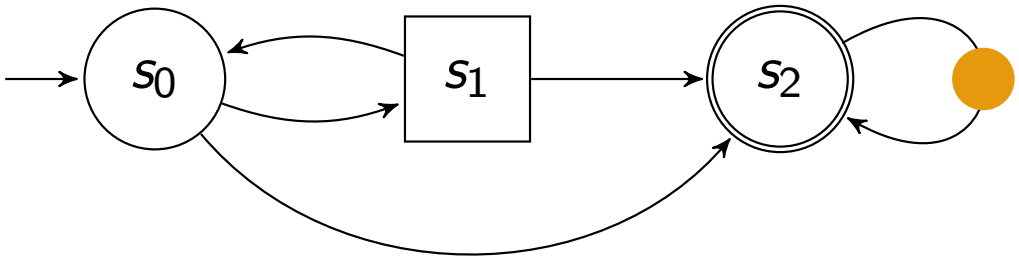
- Modèlisent l'interaction entre deux joueurs : le système (○) et son adversaire, l'environnement (□).



- ▷ Les parties sont infinies. Elles sont déclarées *gagnantes* pour le système si elles *satisfont la spécification* (= comportement acceptable). Sinon, l'environnement gagne.
- ▷ P.ex. s_2 doit être visité infiniment souvent.

Jeux sur graphes

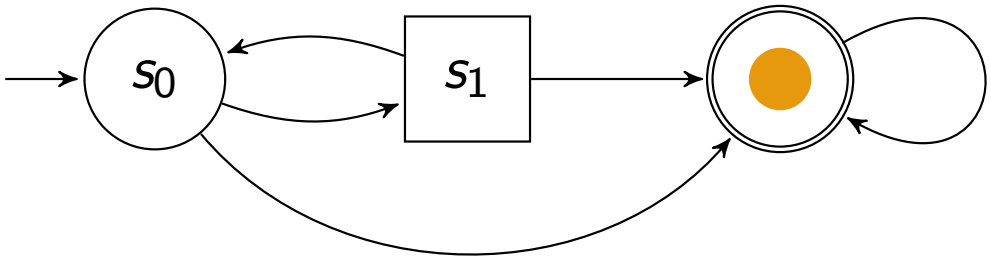
- Modèlisent l'interaction entre deux joueurs : le système (○) et son adversaire, l'environnement (□).



- ▷ Les parties sont infinies. Elles sont déclarées *gagnantes* pour le système si elles *satisfont la spécification* (= comportement acceptable). Sinon, l'environnement gagne.
- ▷ P.ex. s_2 doit être visité infiniment souvent.

Jeux sur graphes

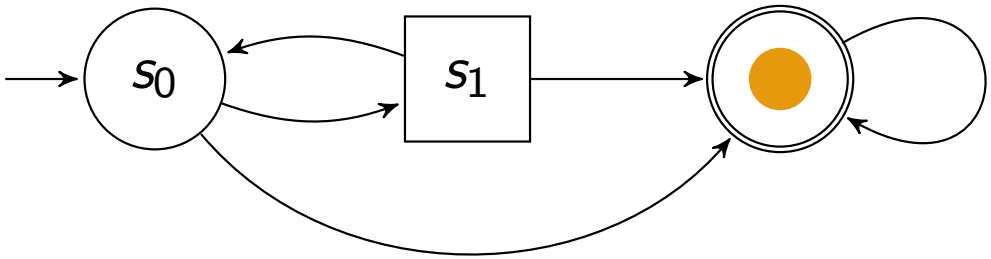
- Modèlisent l'interaction entre deux joueurs : le système (○) et son adversaire, l'environnement (□).



- ▷ Les parties sont infinies. Elles sont déclarées *gagnantes* pour le système si elles *satisfont la spécification* (= comportement acceptable). Sinon, l'environnement gagne.
- ▷ P.ex. s_2 doit être visité infiniment souvent.

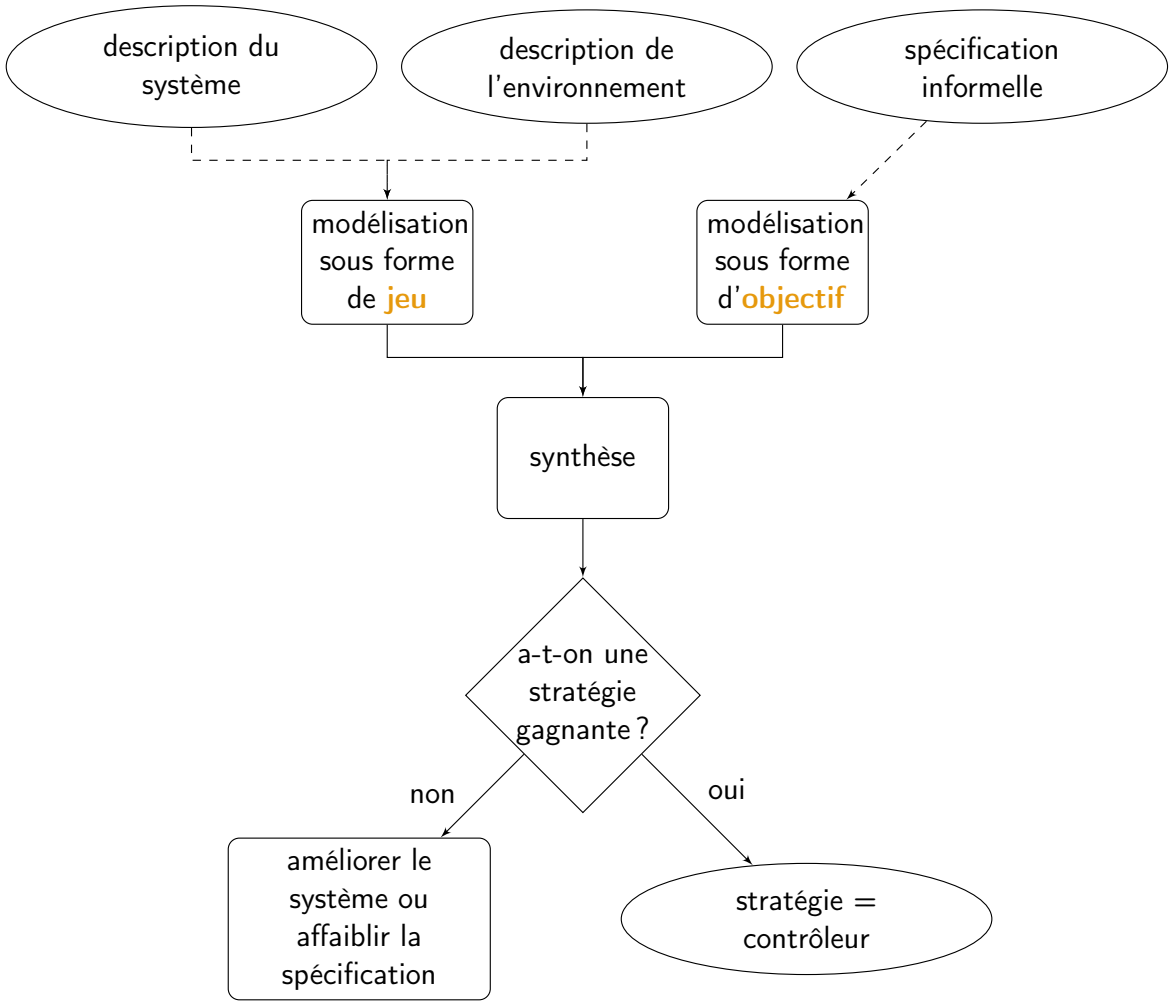
Jeux sur graphes

- Modèlisent l'interaction entre deux joueurs : le système (○) et son adversaire, l'environnement (□).

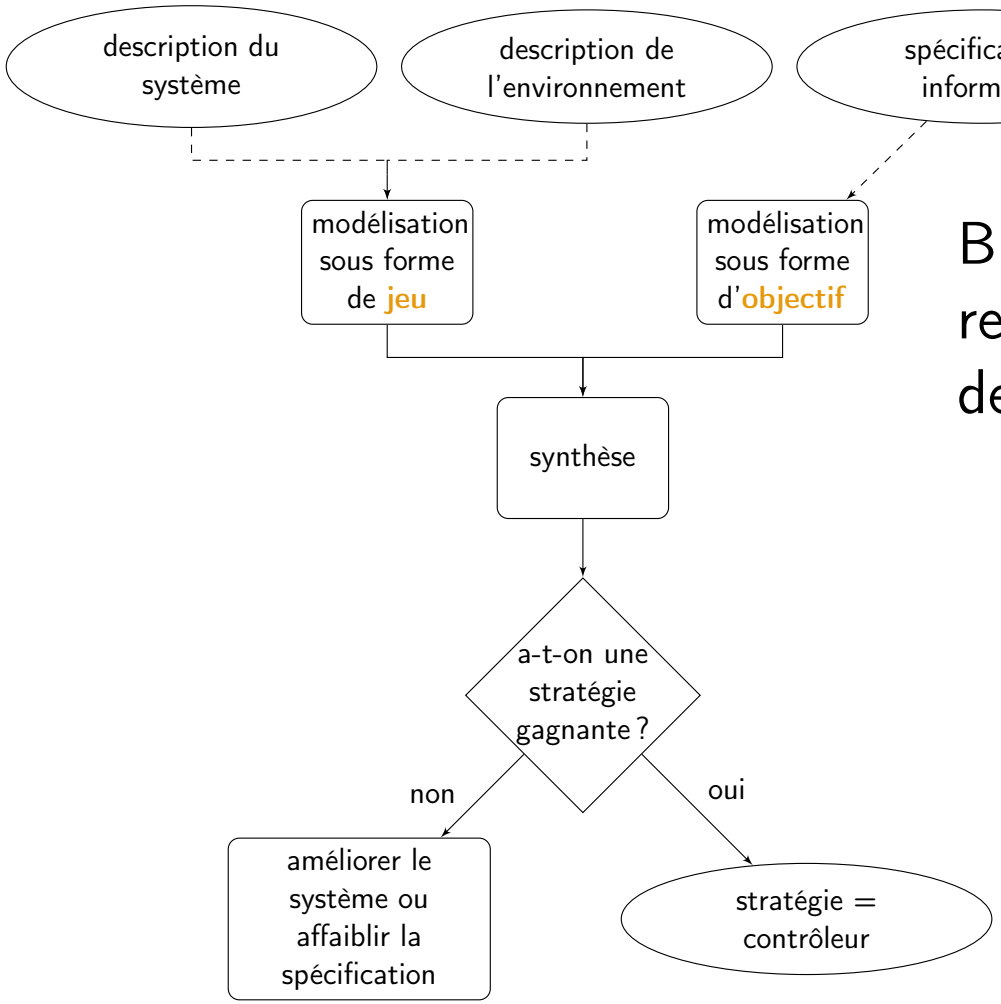


- ▷ Un système fiable doit gagner *contre toute stratégie de son environnement*.
- ▷ **Trouver une stratégie gagnante = synthétiser un contrôleur fiable.**

Synthèse via les jeux sur graphes



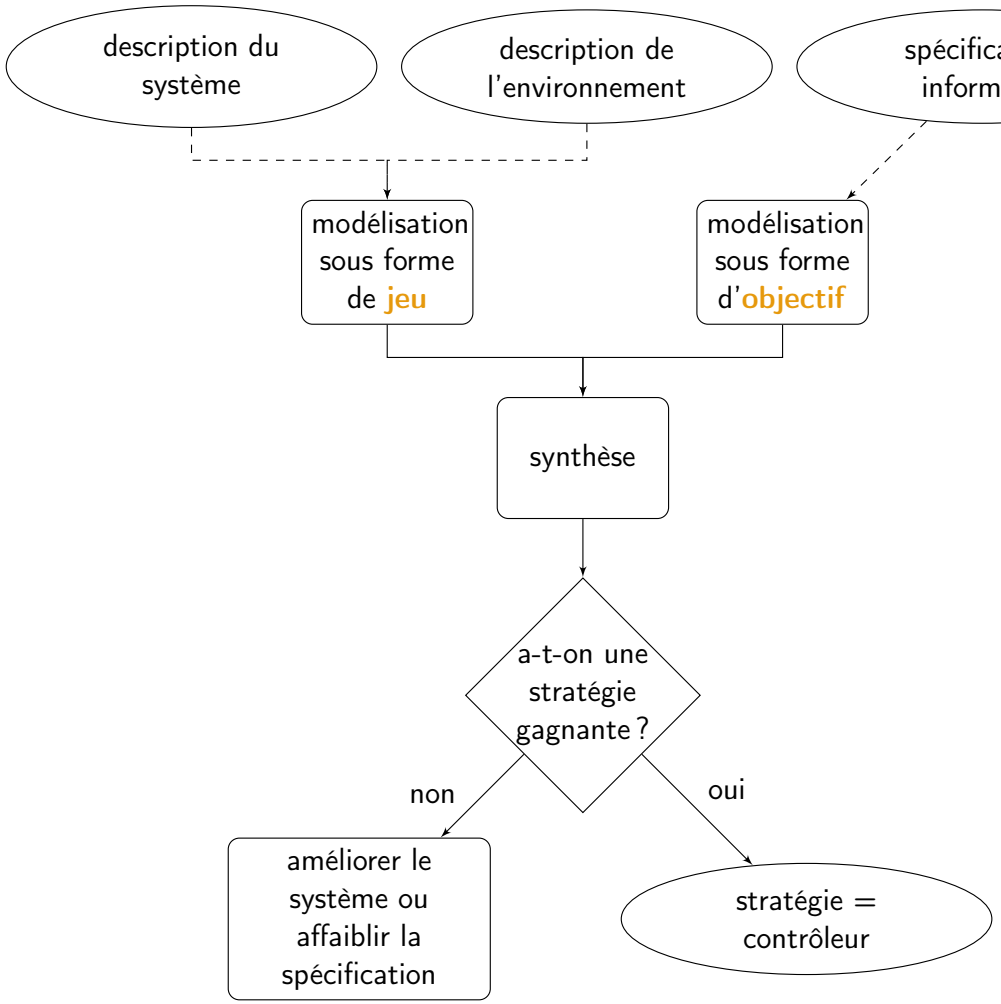
Synthèse via les jeux sur graphes



But de cette thèse : étudier différentes classes de jeux, d'objectifs, de stratégies

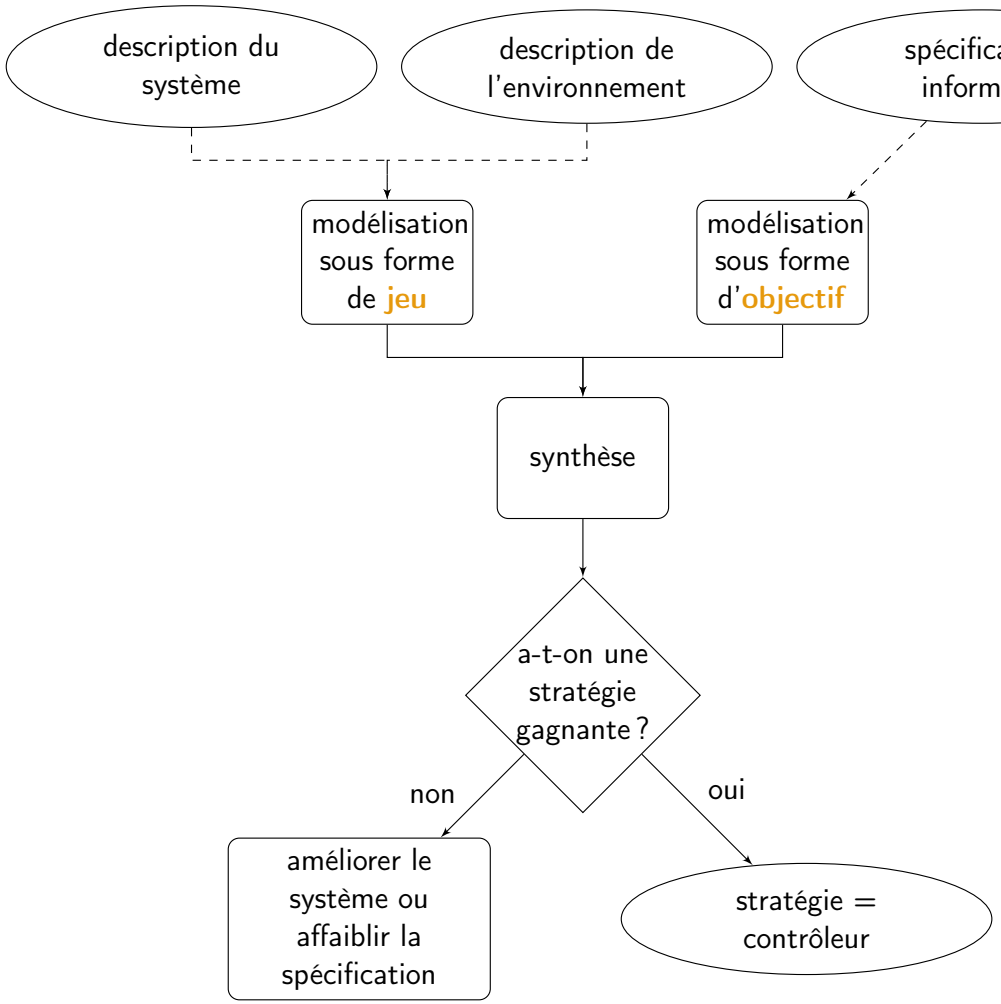
- ▷ expressivité vs. complexité
- ▷ étude fondamentale
- ▷ développement d'algorithmes efficaces

Synthèse via les jeux sur graphes



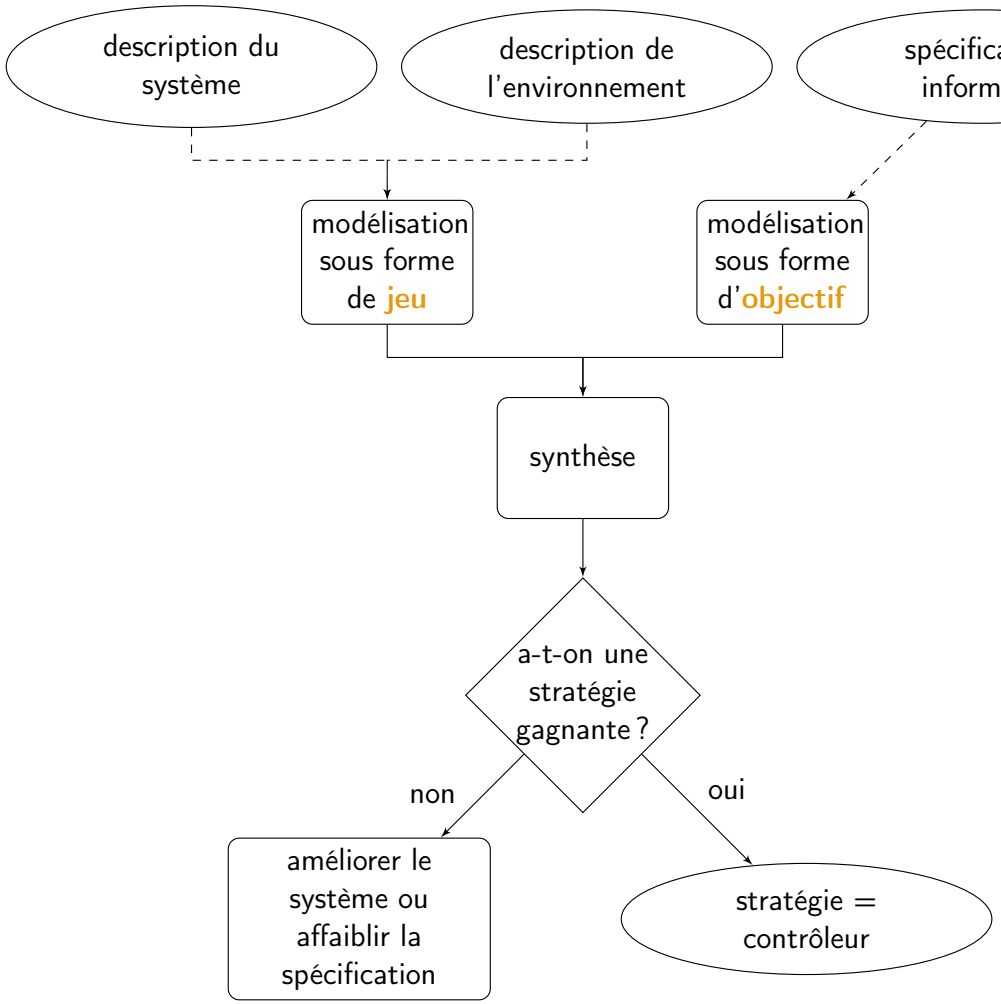
1. Un joueur peut-il garantir la victoire ?

Synthèse via les jeux sur graphes



1. Un joueur peut-il garantir la victoire ?
2. Peut-on décider lequel ?

Synthèse via les jeux sur graphes



1. Un joueur peut-il garantir la victoire ?
2. Peut-on décider lequel ?
3. Quelle est la complexité nécessaire pour une stratégie gagnante ?

Besoin de modèles riches

Nous travaillons sur des **modèles mathématiques abstraits**

- ▷ pour pouvoir exprimer une spécification donnée, il faut que nos modèles soient suffisamment puissants.

Besoin de modèles riches

Nous travaillons sur des **modèles mathématiques abstraits**

- ▷ pour pouvoir exprimer une spécification donnée, il faut que nos modèles soient suffisamment puissants.

Qu'en est-il ?

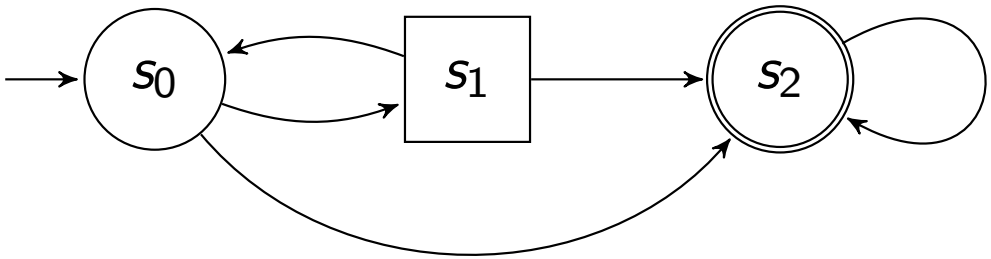
Jeux qualitatifs (Booléens)

- Le modèle traditionnel [GTW02].
- Une partie correspond à un comportement acceptable ou non, sans interprétation de la qualité relative du comportement (performance).

Jeux qualitatifs (Booléens)

- Le modèle traditionnel [GTW02].
- Une partie correspond à un comportement acceptable ou non, sans interprétation de la qualité relative du comportement (performance).

▷ Exemple : objectif de Büchi, visiter s_2 de manière répétée.



▷ En général, des propriétés fonctionnelles (p.ex. toujours répondre aux requêtes).

Jeux qualitatifs (Booléens)

- Le modèle traditionnel [GTW02].
- Une partie correspond à un comportement acceptable ou non, sans interprétation de la qualité relative du comportement (performance).

▷ Des **stratégies gagnantes simples** existent : sans mémoire, sans aléatoire.

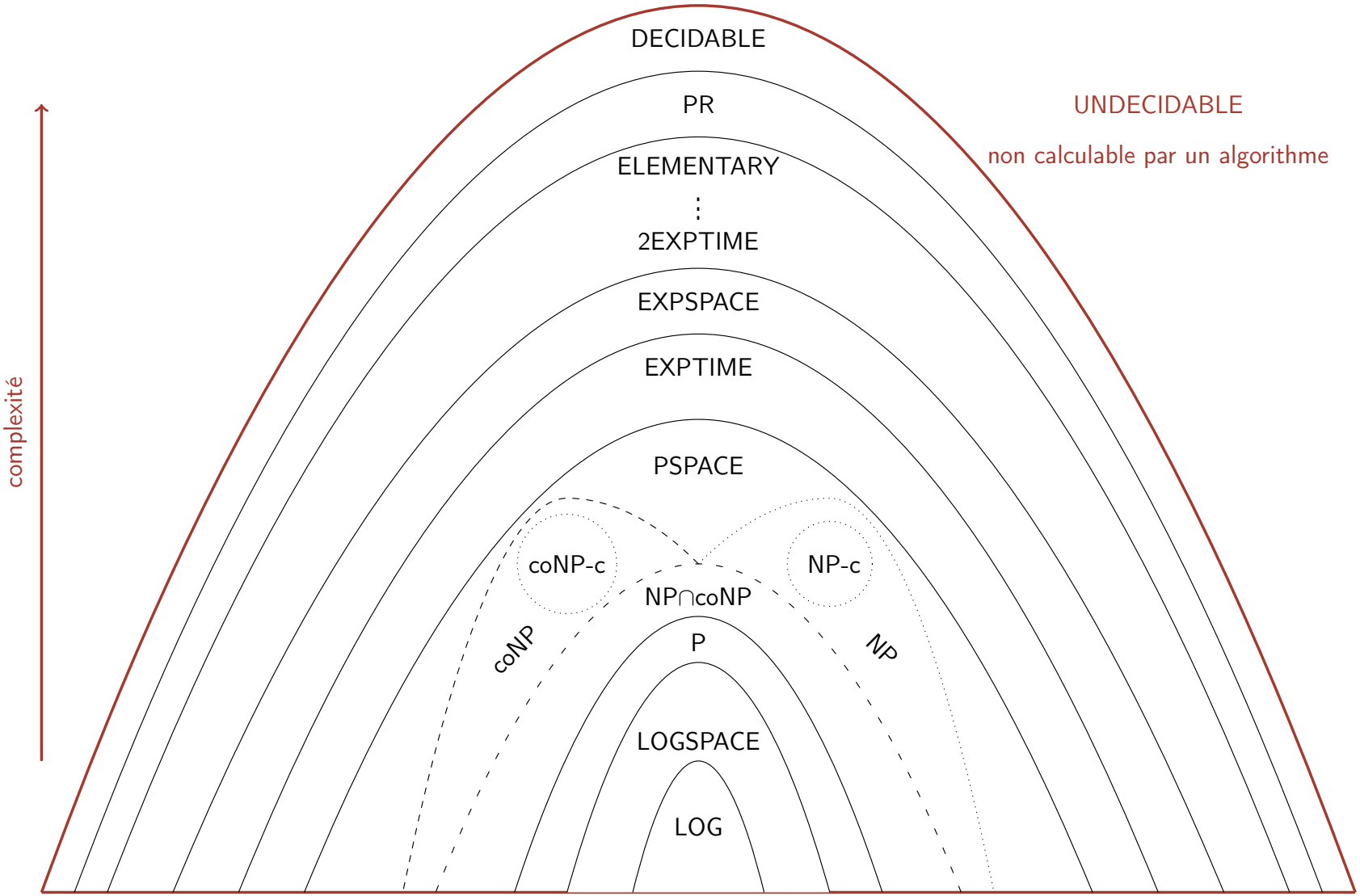
Jeux qualitatifs (Booléens)

- Le modèle traditionnel [GTW02].
- Une partie correspond à un comportement acceptable ou non, sans interprétation de la qualité relative du comportement (performance).

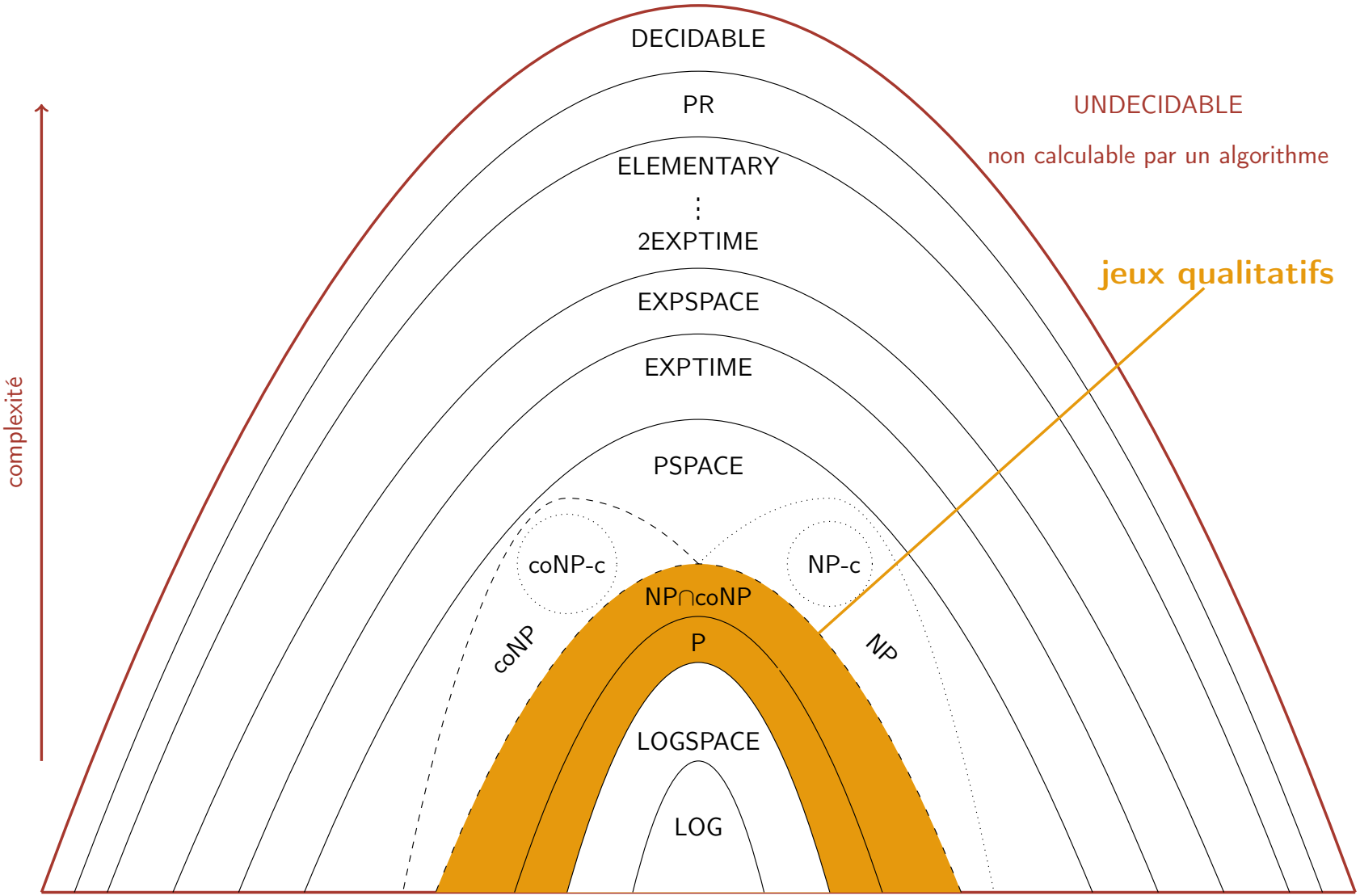
▷ Des **stratégies gagnantes simples** existent : sans mémoire, sans aléatoire.

▷ **Complexité algorithmique faible** : des algorithmes efficaces existent.

Complexité algorithmique



Complexité algorithmique



Jeux quantitatifs

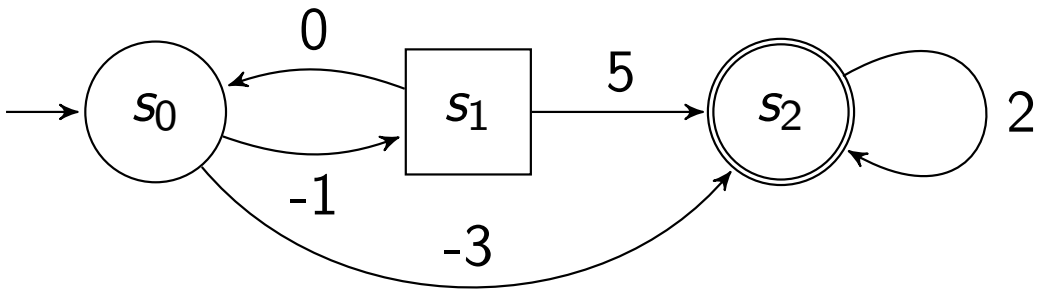
- Focus actuel de la communauté.
- Contraintes de ressources ou de performance (consommation d'énergie, temps de réponse) [CdAHS03, BCHJ09].

Jeux quantitatifs

- Focus actuel de la communauté.
- Contraintes de ressources ou de performance (consommation d'énergie, temps de réponse) [CdAHS03, BCHJ09].

▷ On associe des poids aux transitions et on considère des fonctions donnant une **valeur numérique** aux parties.

▷ Par exemple, le temps moyen par action.



Jeux quantitatifs

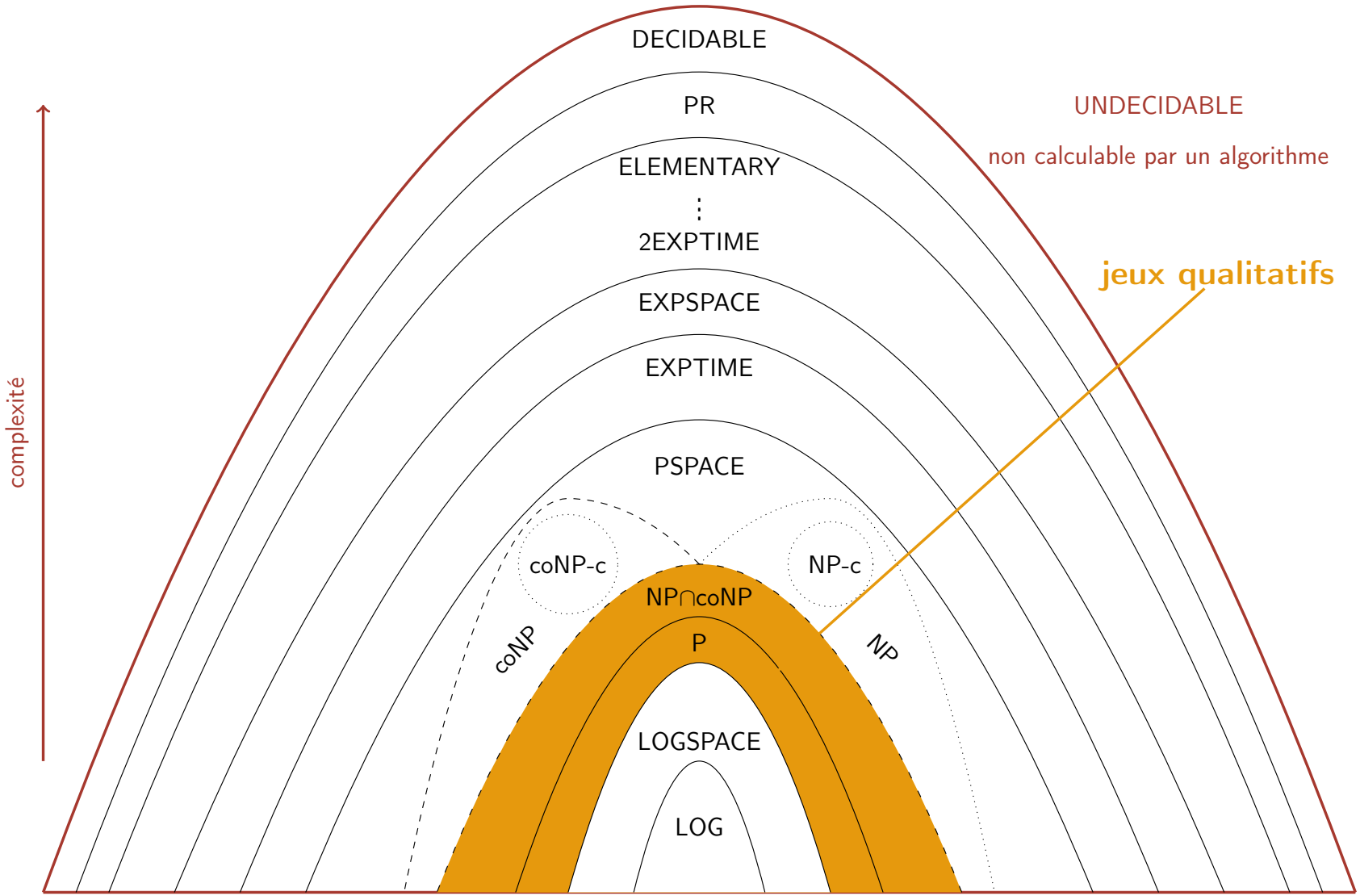
- Focus actuel de la communauté.
- Contraintes de ressources ou de performance (consommation d'énergie, temps de réponse) [CdAHS03, BCHJ09].

▷ On cherche des stratégies assurant un **niveau de performance requis** face à toute stratégie de l'environnement.

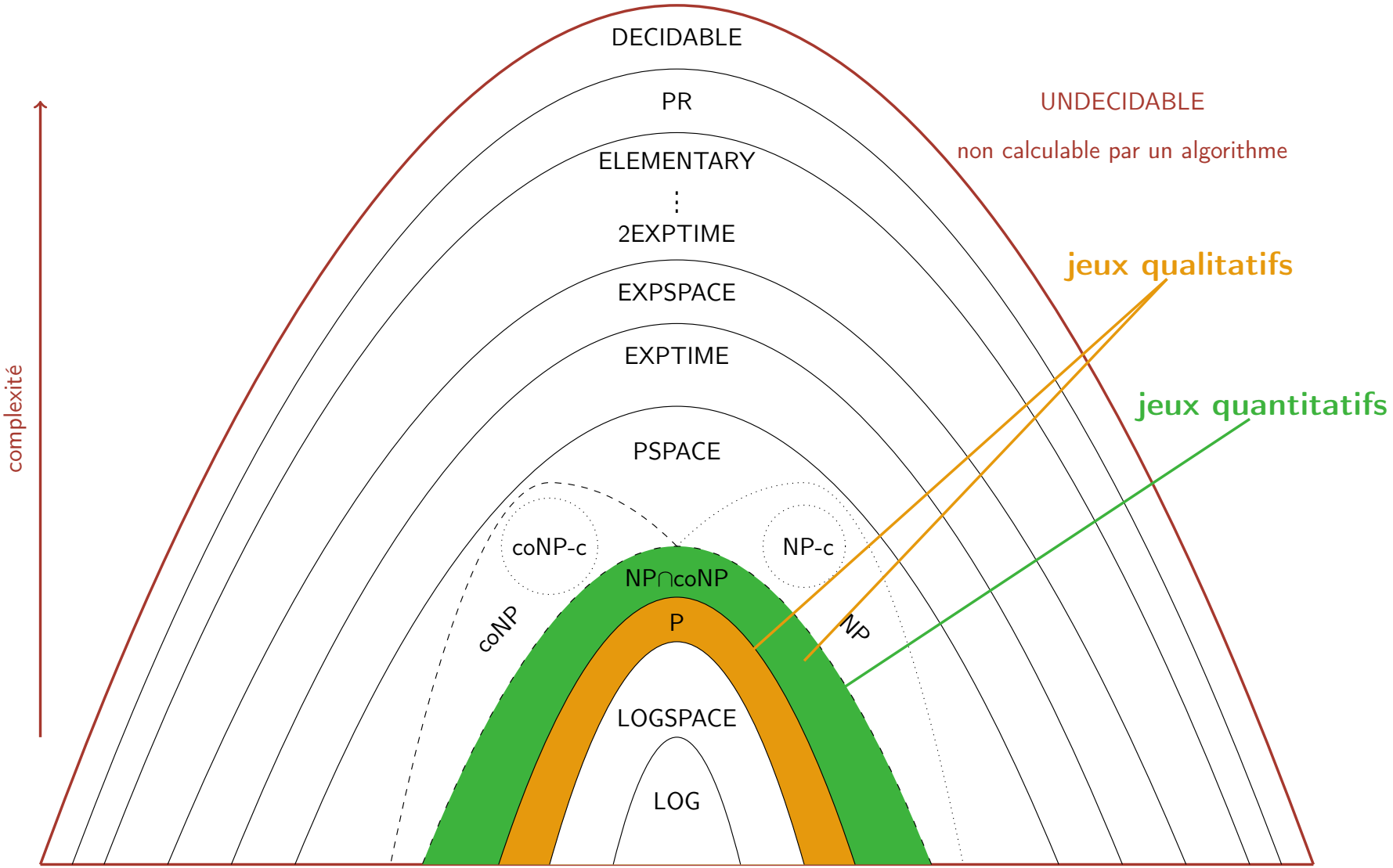
▷ Des **stratégies gagnantes simples** existent : sans mémoire, sans aléatoire.

▷ **Complexité algorithmique légèrement plus élevée** : importantes questions ouvertes (aussi pour certains qualitatifs).

Jeux quantitatifs



Jeux quantitatifs



Jeux quantitatifs multi-critères

- La prochaine étape : modéliser les interactions et compromis entre différents aspects quantitatifs.

▷ Impossible dans la plupart des modèles actuels.

Jeux quantitatifs multi-critères

- La prochaine étape : modéliser les interactions et compromis entre différents aspects quantitatifs.

▷ Impossible dans la plupart des modèles actuels.

▷ Diminuer le temps de réponse d'un serveur peut demander d'augmenter sa puissance et donc sa consommation électrique.

Jeux quantitatifs multi-critères

- La prochaine étape : modéliser les interactions et compromis entre différents aspects quantitatifs.

▷ Impossible dans la plupart des modèles actuels.

▷ Diminuer le temps de réponse d'un serveur peut demander d'augmenter sa puissance et donc sa consommation électrique.

Le but de cette thèse est d'étudier et de développer de tels modèles multi-critères.

Jeux quantitatifs multi-critères

- La prochaine étape : modéliser les interactions et compromis entre différents aspects quantitatifs.

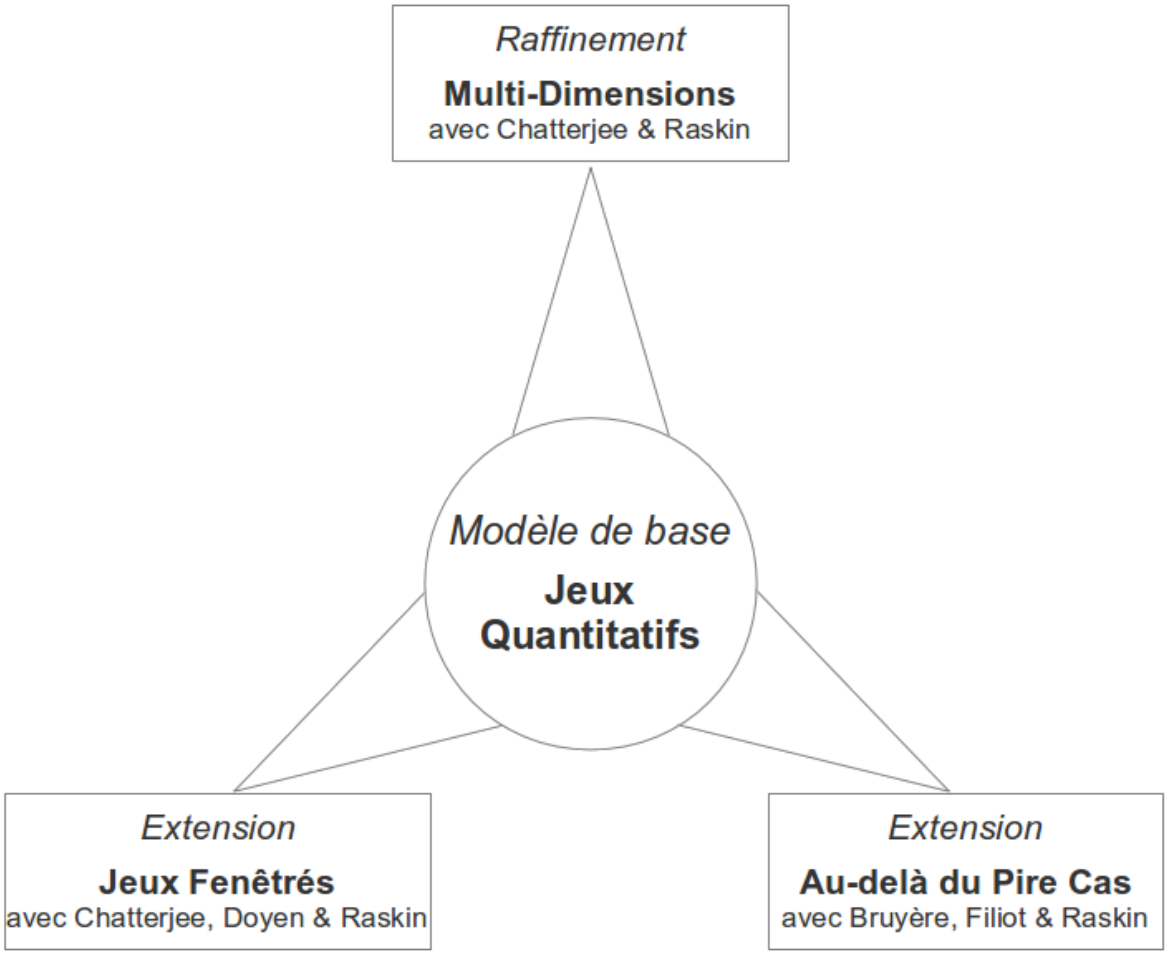
▷ Impossible dans la plupart des modèles actuels.

▷ Diminuer le temps de réponse d'un serveur peut demander d'augmenter sa puissance et donc sa consommation électrique.

Le but de cette thèse est d'étudier et de développer de tels modèles multi-critères.

Tout devient vite beaucoup plus complexe !
Stratégies et complexité algorithmique.

Contributions



Transitions des modèles mono-critères vers les modèles **multi-critères**.

Exemple : tondeuse robotisée

- But : synthétiser un contrôleur pour une tondeuse robotisée.

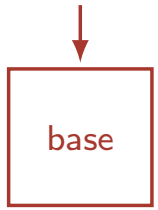
- Multi-critères :
  - ▷ objectif qualitatif,
  - ▷ plusieurs objectifs quantitatifs.

[Ran13] M. Randour. Automated synthesis of reliable and efficient systems through game theory : a case study. ECCS'12, Springer Proceedings in Complexity XVII, 8 pages, Springer, 2013.

Modélisation sous forme de jeu

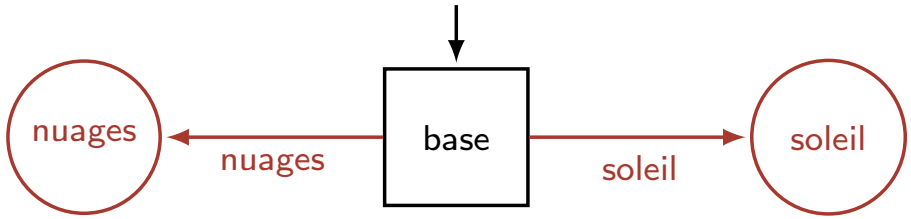
- ▷ Interactions entre la tondeuse et son environnement vues comme un jeu.
- ▷ Modélisation de la spécification sous forme d'objectif pour la tondeuse.

Modélisation sous forme de jeu



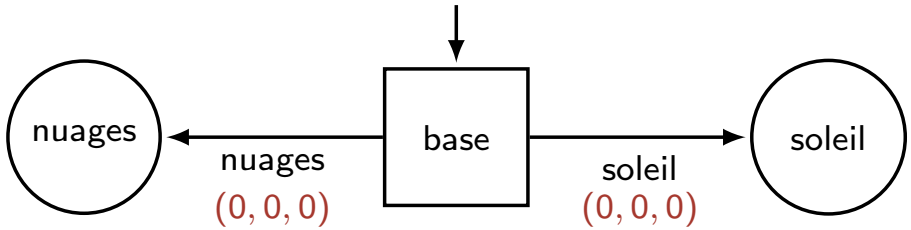
- La tondeuse démarre le jeu sur sa base.

Modélisation sous forme de jeu



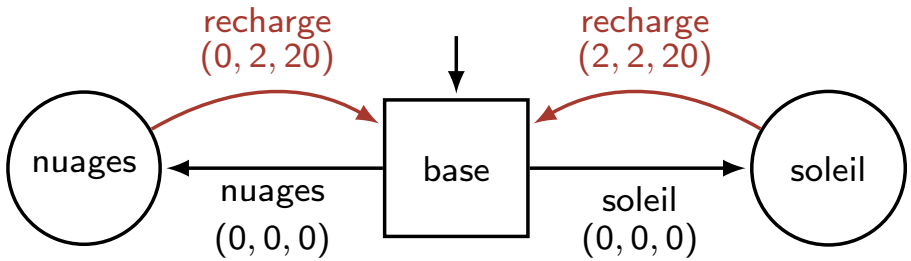
- La météo peut être nuageuse ou ensoleillée.

Modélisation sous forme de jeu



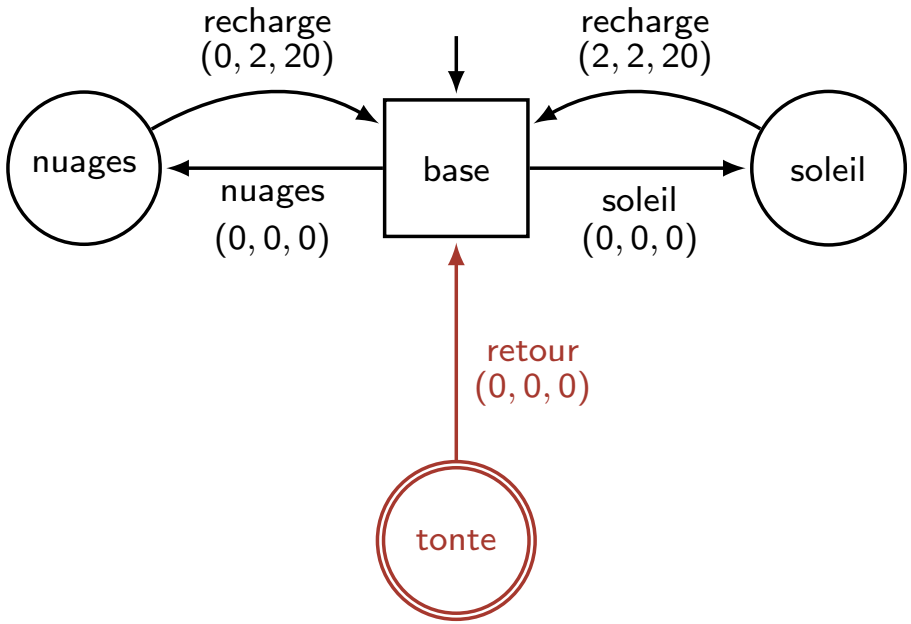
- ▷ Batterie rechargée par le soleil grâce à des panneaux solaires. Réservoir rechargé à la base. Les deux sont infinis.
- ▷ Chaque action prend du temps.

Modélisation sous forme de jeu



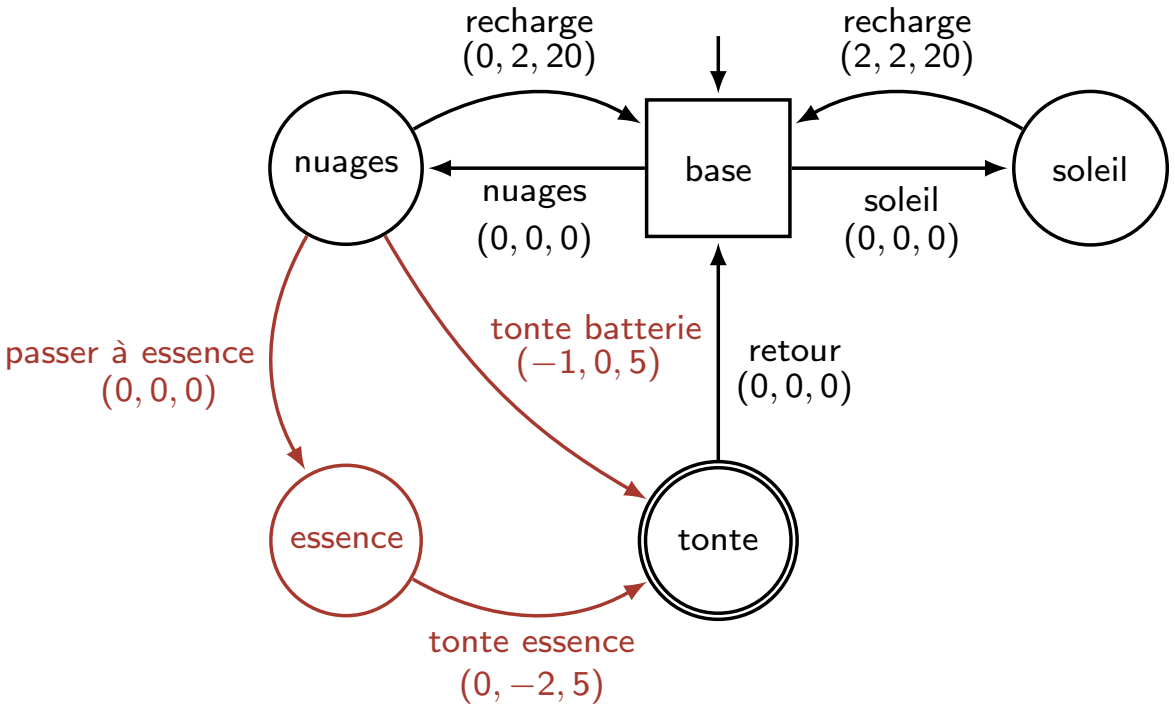
- Batterie rechargée uniquement si ensoleillé (2 unités).
- Recharge du réservoir (2 unités) quelque soit la météo.
- Recharger prend 20 unités de temps.

Modélisation sous forme de jeu



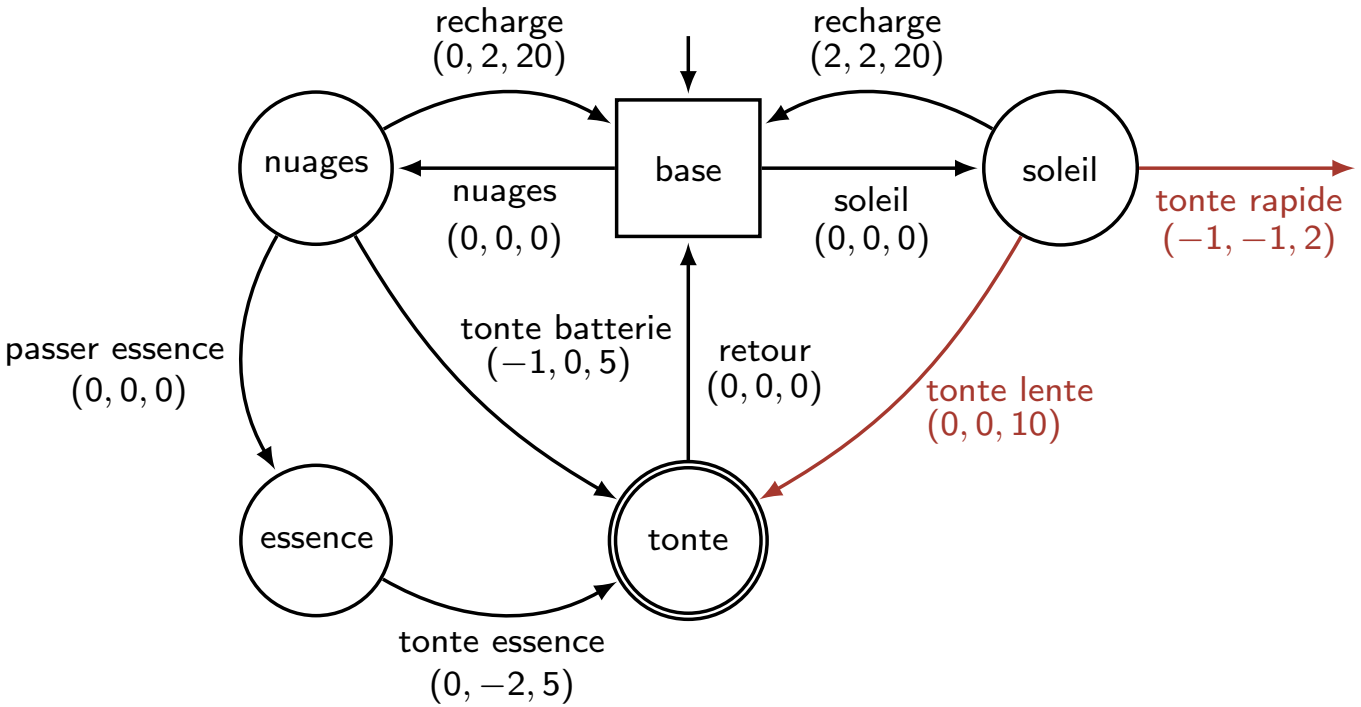
- ▷ Pas de contrainte sur la fréquence de tonte.
- ▷ Mais l'herbe ne doit pas pousser sans limite $\rightsquigarrow$ la tondeuse doit tondre infiniment souvent.

Modélisation sous forme de jeu



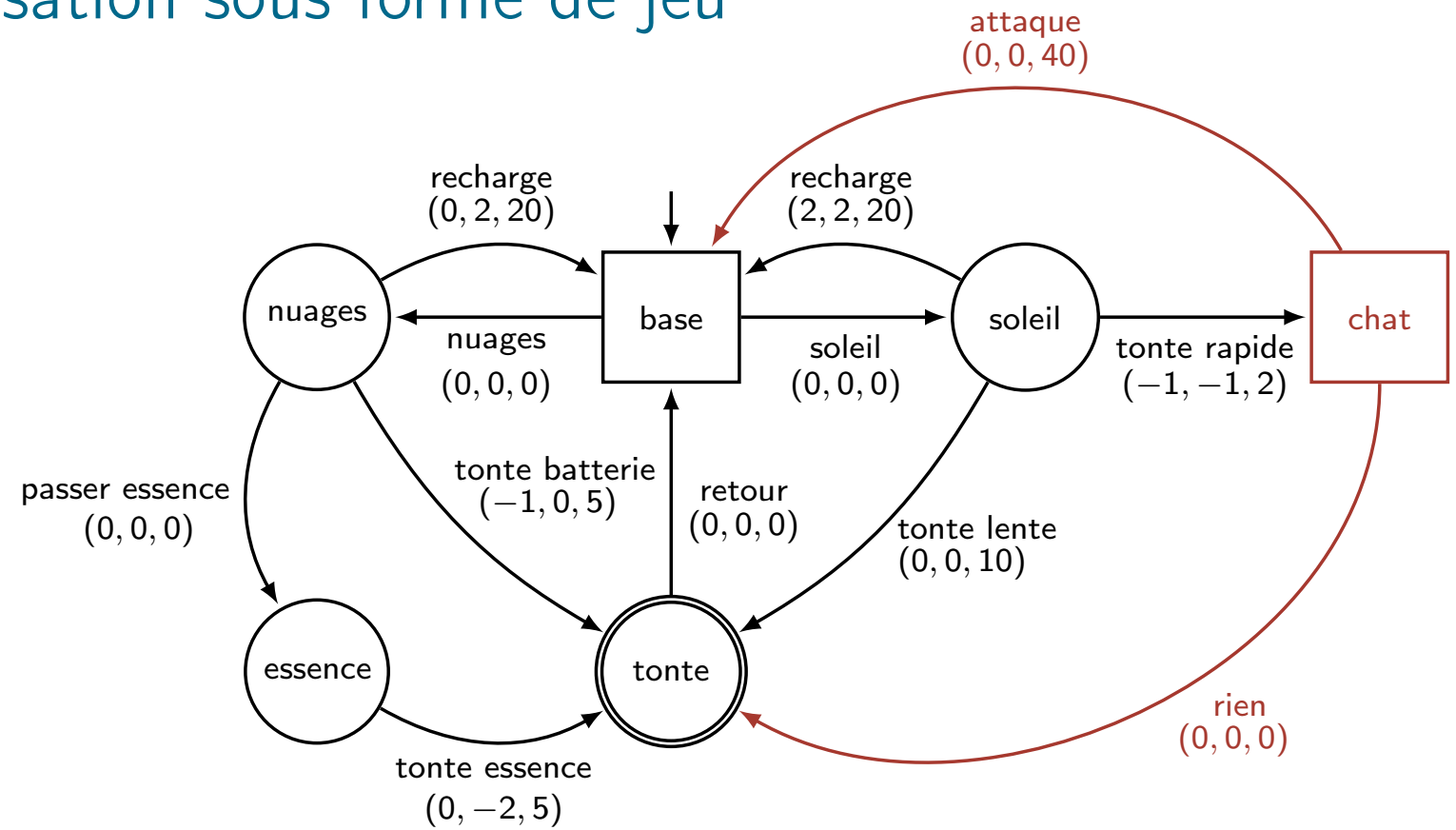
- Quand nuageux, tonte sous batterie (1 unité) ou sous essence (2 unités).
- Même vitesse (5 unités de temps).

Modélisation sous forme de jeu



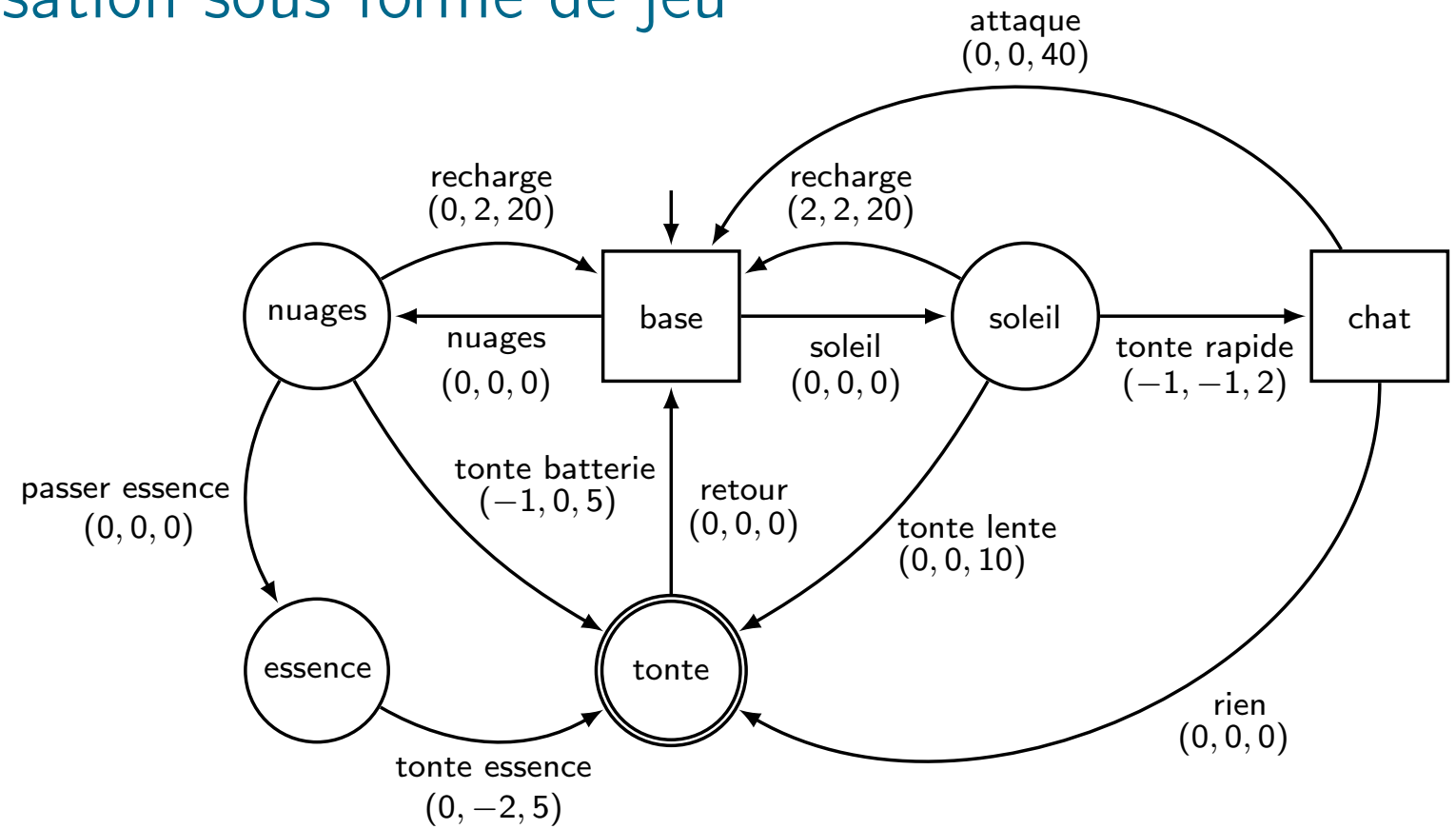
- Quand ensoleillé, la tonte lente ne consomme rien mais prend 10 unités de temps.
- La tonte rapide consomme 1 unité d'essence et 1 unité de batterie mais ne prend que 2 unités de temps.

Modélisation sous forme de jeu



- ▷ La tonte rapide fait du bruit et peut réveiller le chat
  $\rightsquigarrow$ interruption et 40 unités de temps perdues.
- ▷ Le chat ne sort pas si le temps est maussade.

Modélisation sous forme de jeu



- Quel est l'objectif de la tondeuse, c'est-à-dire la spécification à garantir ?

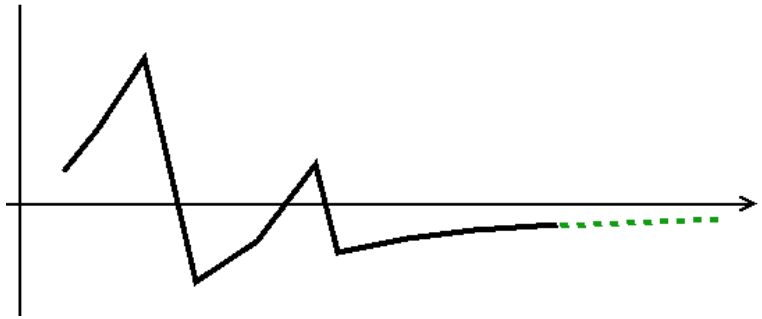
Objectifs de victoire

- *Energie* : l'essence et la batterie ne doivent jamais descendre en dessous de zéro.

Objectifs de victoire

- *Energie* : l'essence et la batterie ne doivent jamais descendre en dessous de zéro.

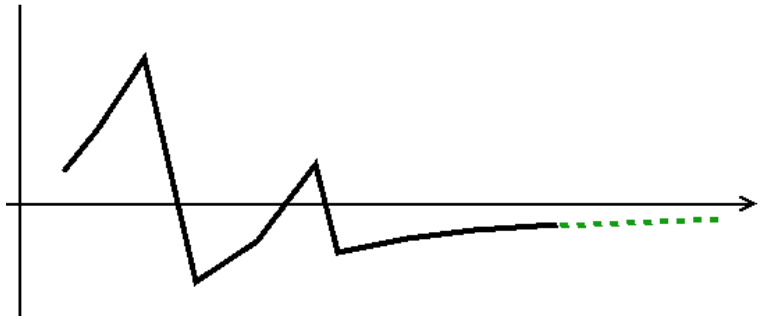
- *Temps moyen* (mean-payoff) : le temps moyen par action doit être ≤ 10.



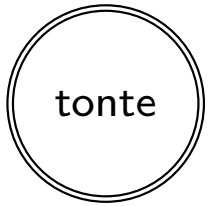
Objectifs de victoire

- *Energie* : l'essence et la batterie ne doivent jamais descendre en dessous de zéro.

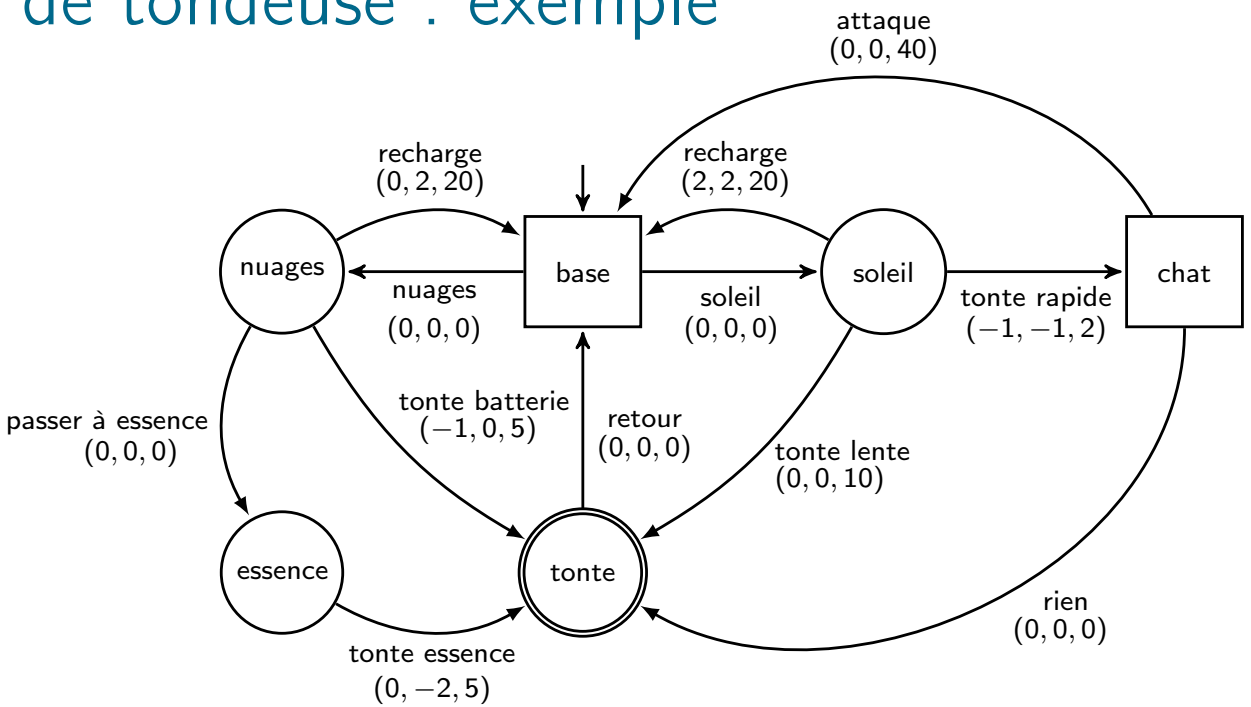
- *Temps moyen* (mean-payoff) : le temps moyen par action doit être ≤ 10.



- *Tonte répétée* : nombre infini de visites au long d'une partie.



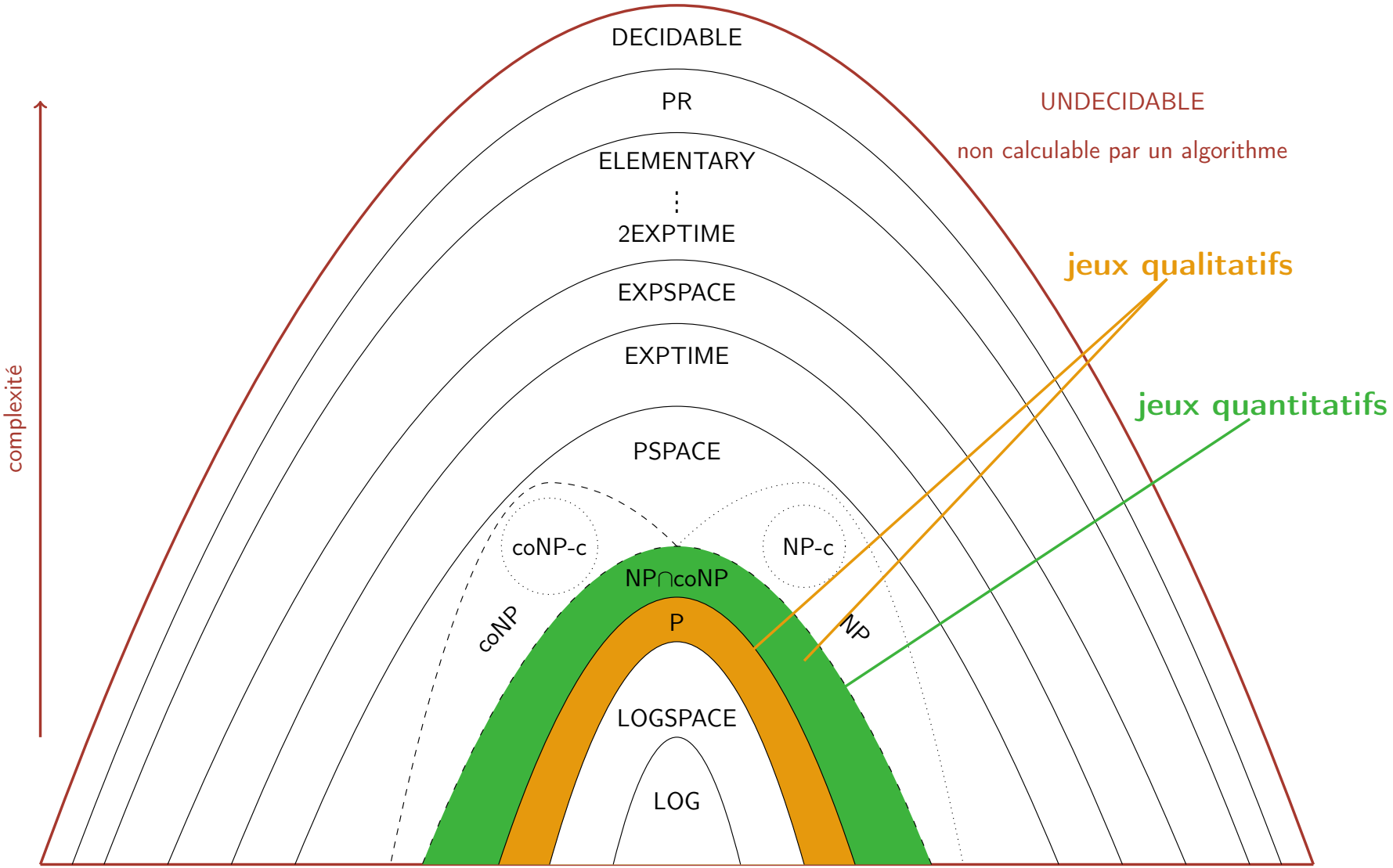
Contrôleur de tondeuse : exemple



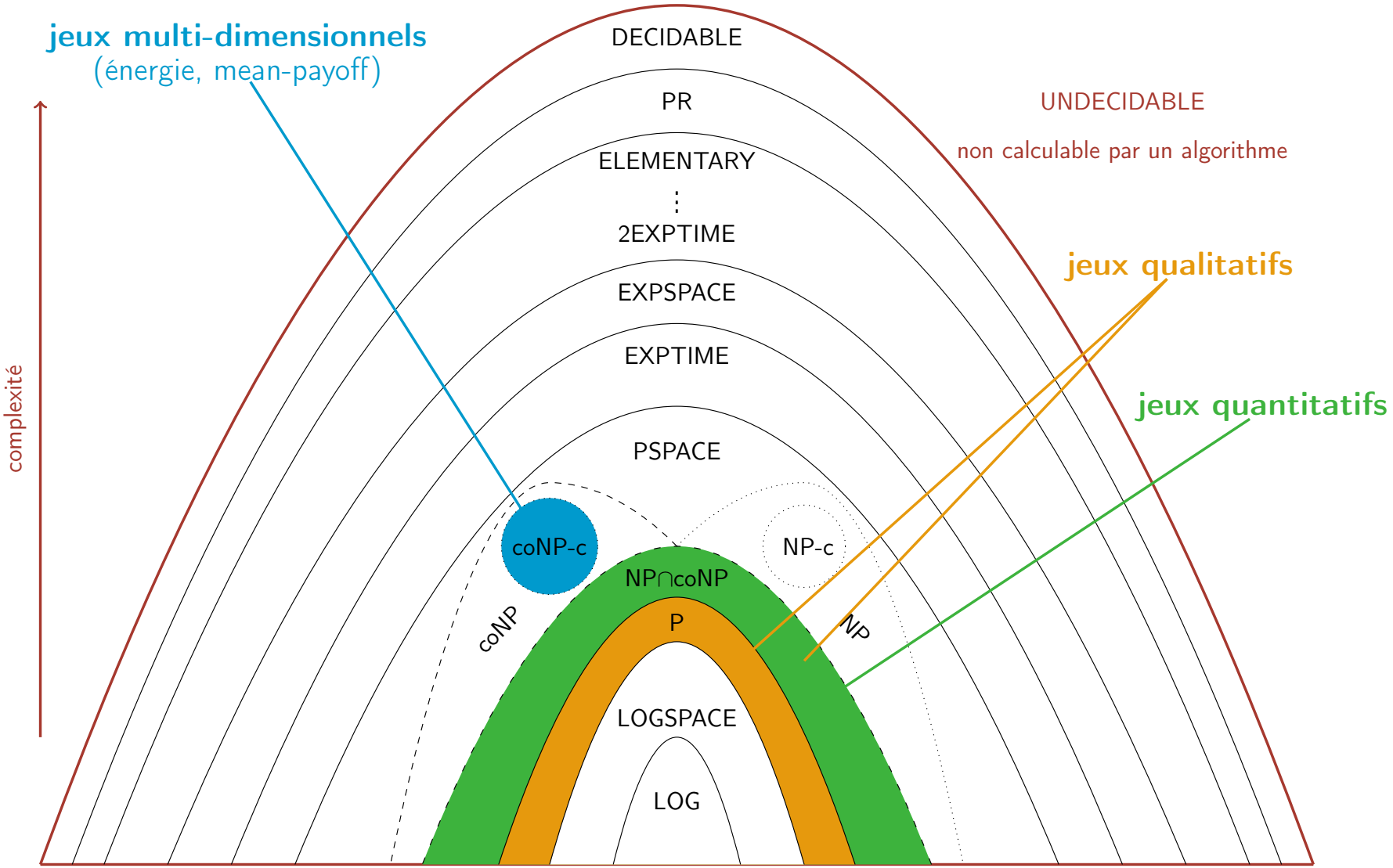
Contrôleur simple (**nécessite de la mémoire**) :

- Démarrer le jeu sans batterie ni essence.
- Si ensoleillé, tondre lentement.
- Si nuageux,
  - ▷ si batterie ≥ 1, tondre sur batterie,
  - ▷ sinon, si essence ≥ 2, tondre sur essence,
  - ▷ sinon, recharger à la base.

Jeux multi-dimensionnels



Jeux multi-dimensionnels



Quelques contributions

Résultats [CRR12, CRR13]

- Une **mémoire exponentielle** suffit et est nécessaire pour les stratégies du système.
- **Algorithme de synthèse** optimal et efficace en pratique.

K. Chatterjee

J.-F. Raskin

[CRR12] K. Chatterjee, M. Randour, J.-F. Raskin. Strategy Synthesis for Multi-Dimensional Quantitative Objectives. CONCUR'12, LNCS 7454, 17 pages, Springer, 2012.

[CRR13] K. Chatterjee, M. Randour, J.-F. Raskin. Strategy Synthesis for Multi-Dimensional Quantitative Objectives. A paraître dans Acta Informatica, 35 pages, Springer, accepté en juillet 2013.

Quelques contributions

Certains objectifs deviennent **extrêmement complexes** dans les jeux multi-dimensionnels !

Résultat [CDRR13]

Les jeux multi-dimensionnels de total-payoff sont **indécidables**.

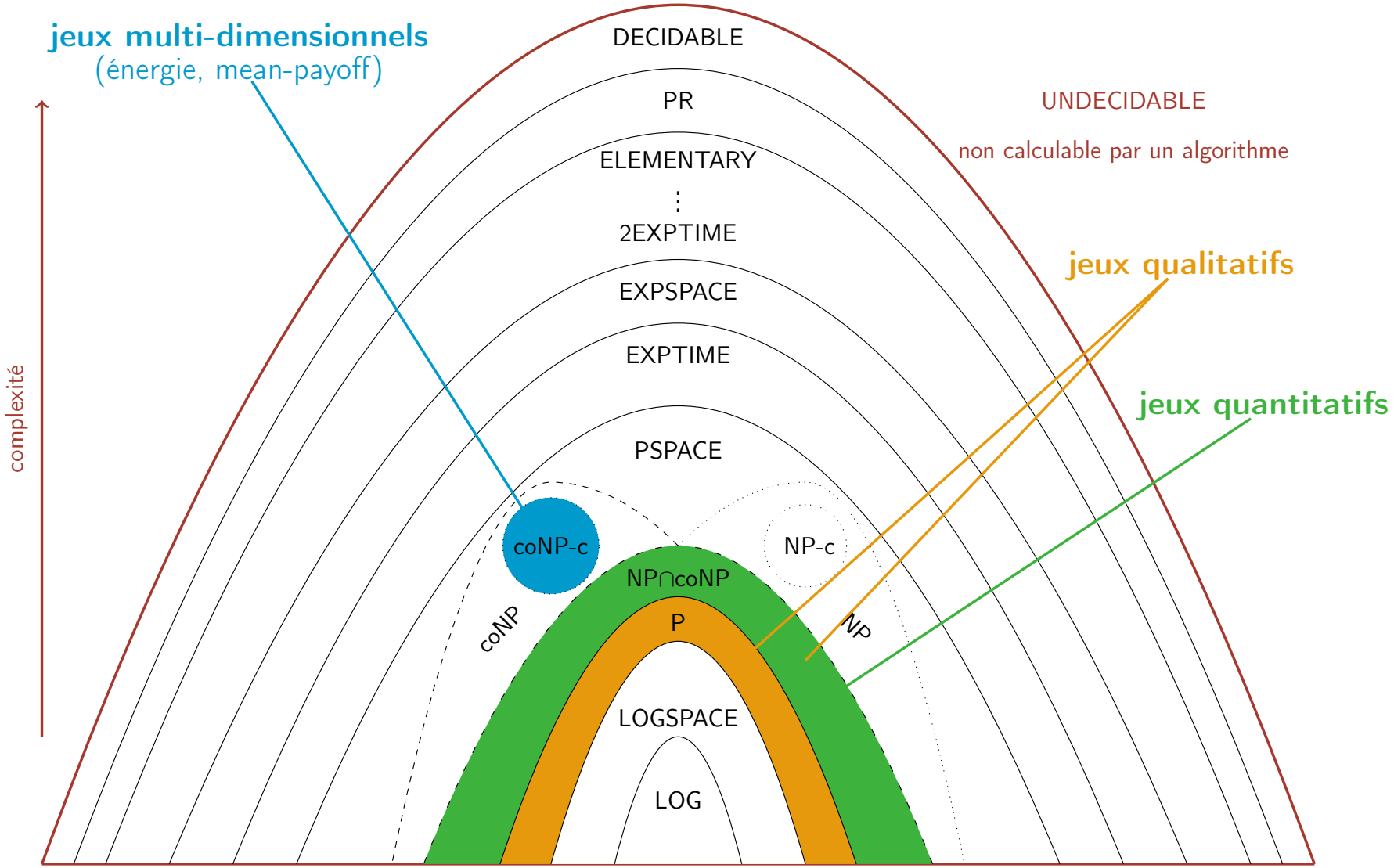
K. Chatterjee

L. Doyen

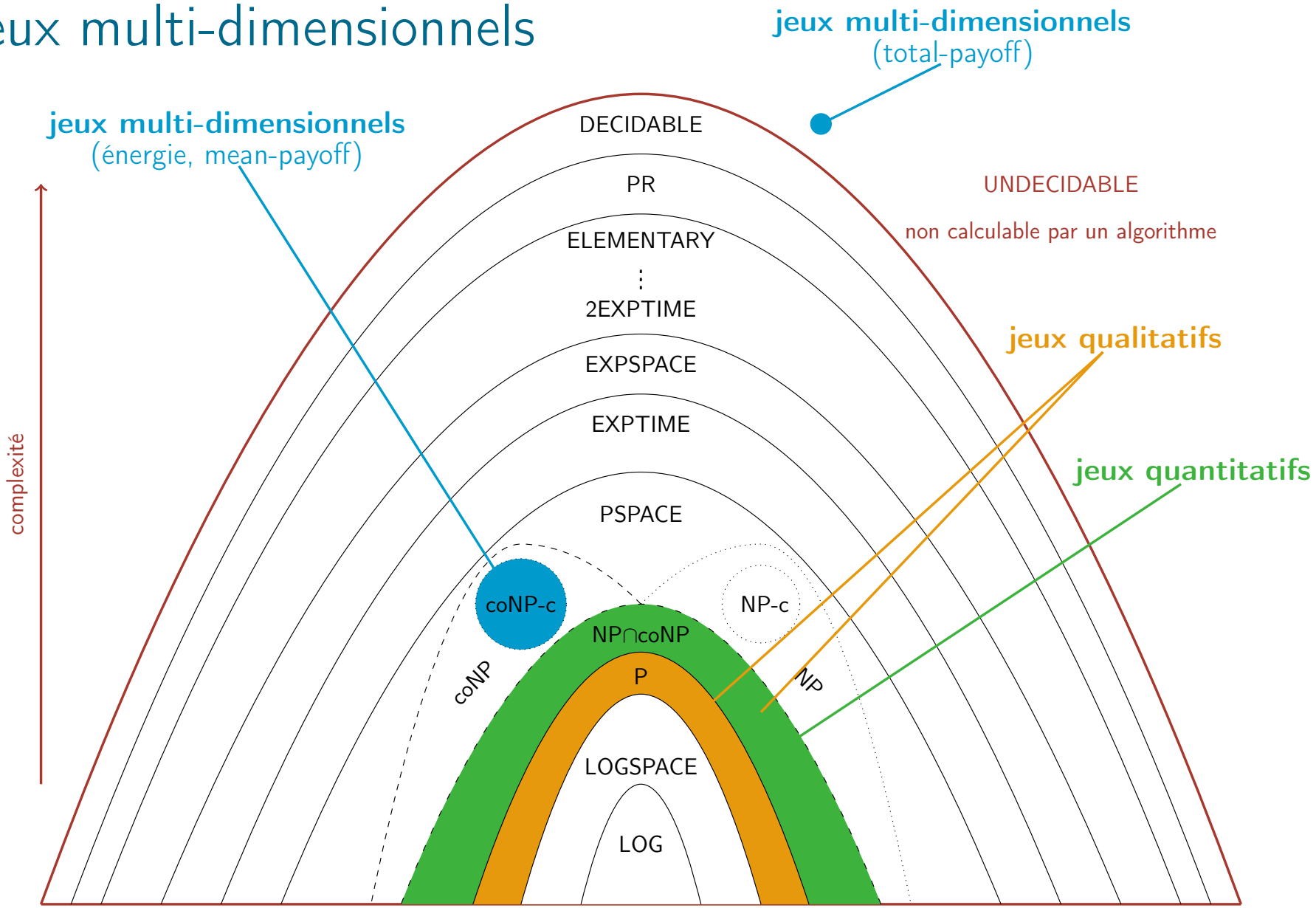
J.-F. Raskin

[CDRR13] K. Chatterjee, L. Doyen, M. Randour, J.-F. Raskin. Looking at Mean-Payoff and Total-Payoff through Windows. ATVA'13, LNCS 8172, 15 pages, Springer, 2013.

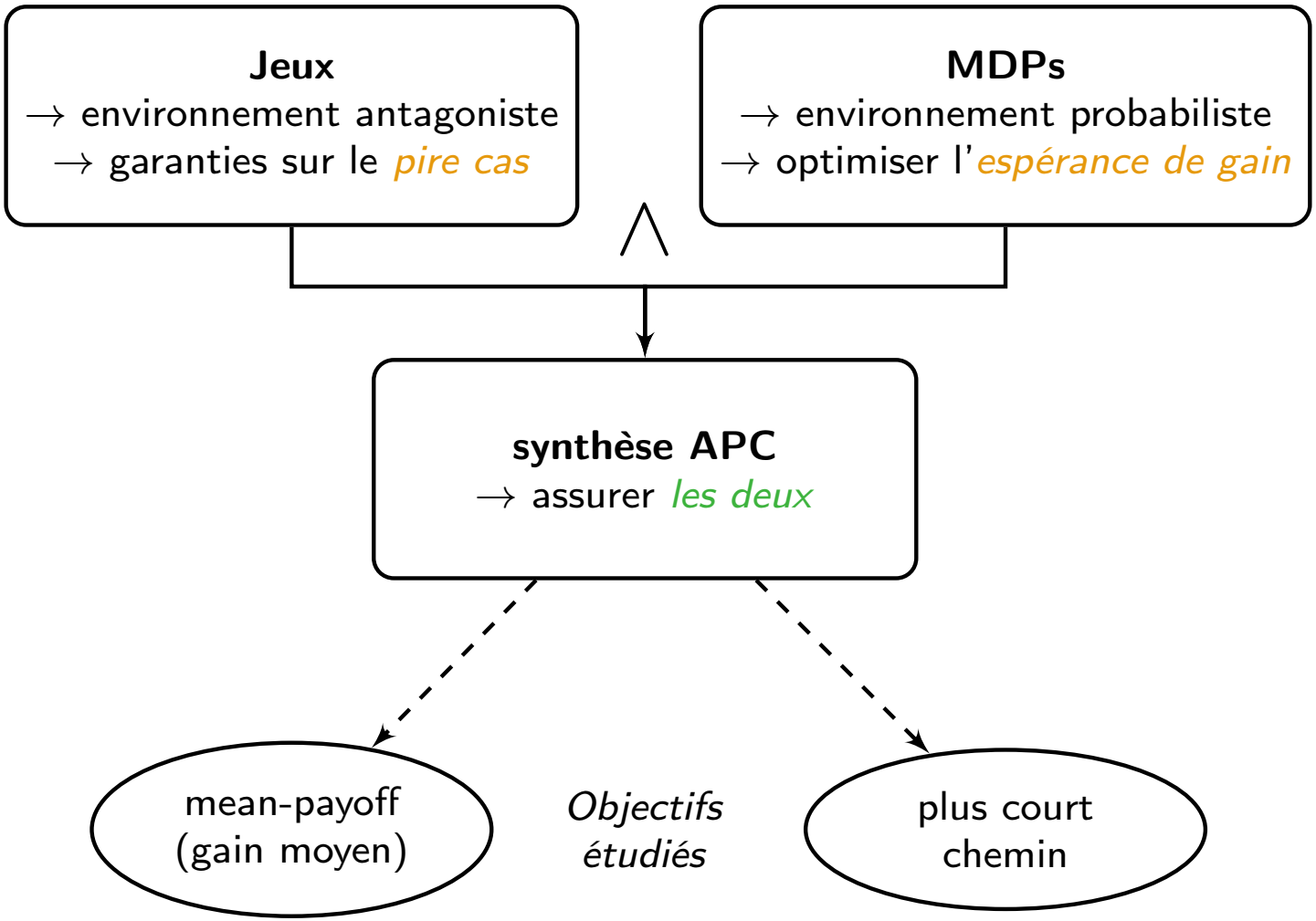
Jeux multi-dimensionnels



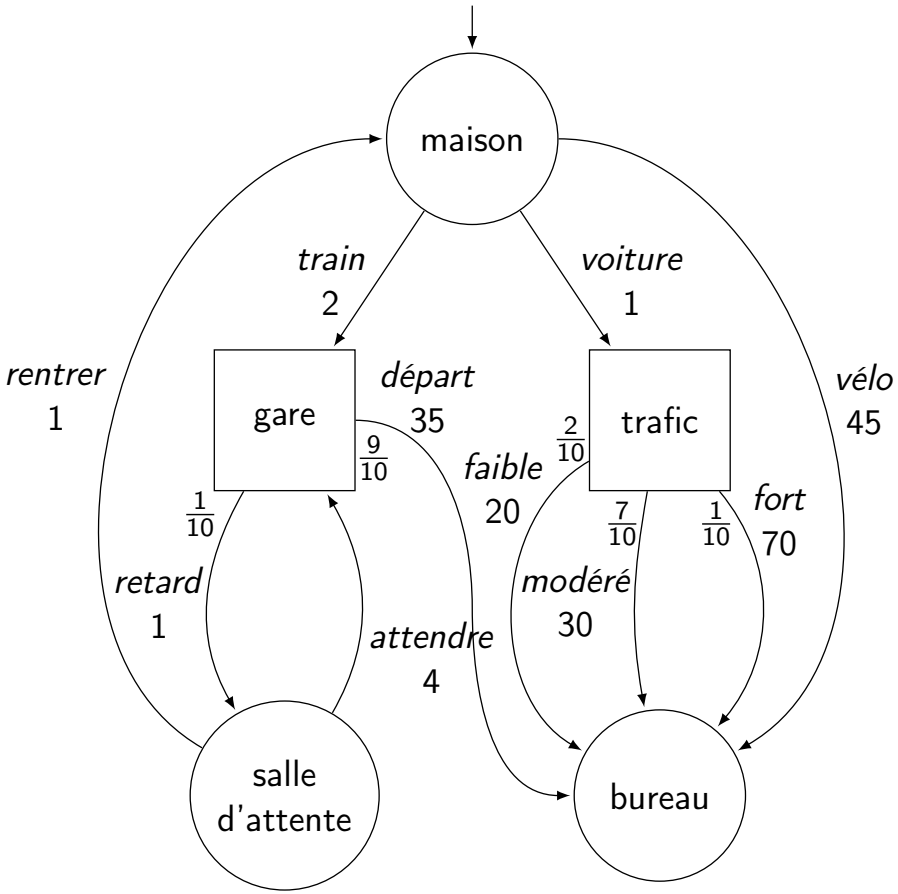
Jeux multi-dimensionnels



Combiner deux modèles classiques

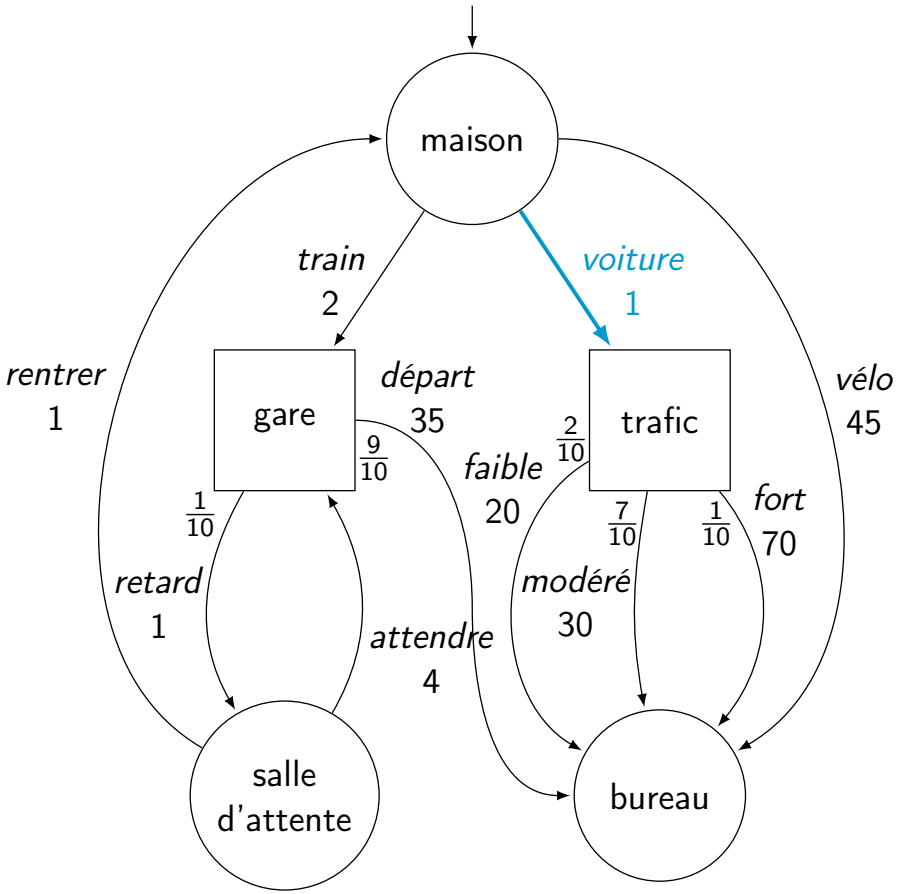


Exemple : aller au bureau



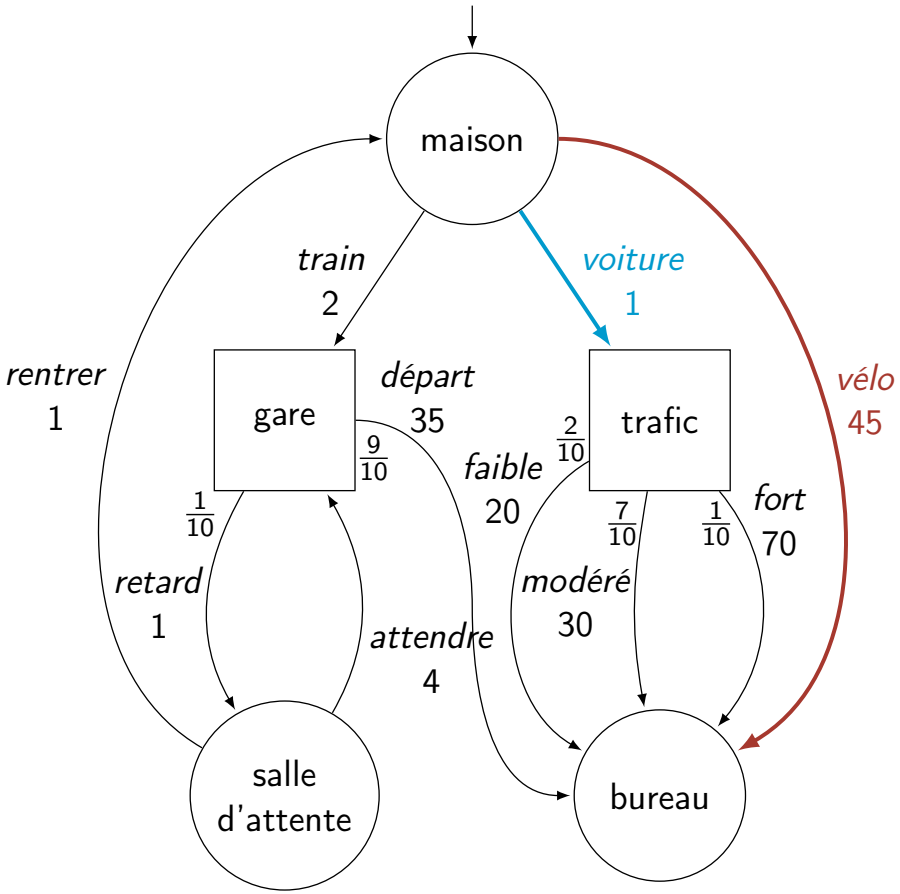
- Poids = minutes
- Objectif : *minimiser le temps espéré ($\sim$ moyen)* pour atteindre le bureau.
- **Mais**, réunion dans une heure ! Besoin de *garanties strictes* sur le pire temps possible.

Exemple : aller au bureau



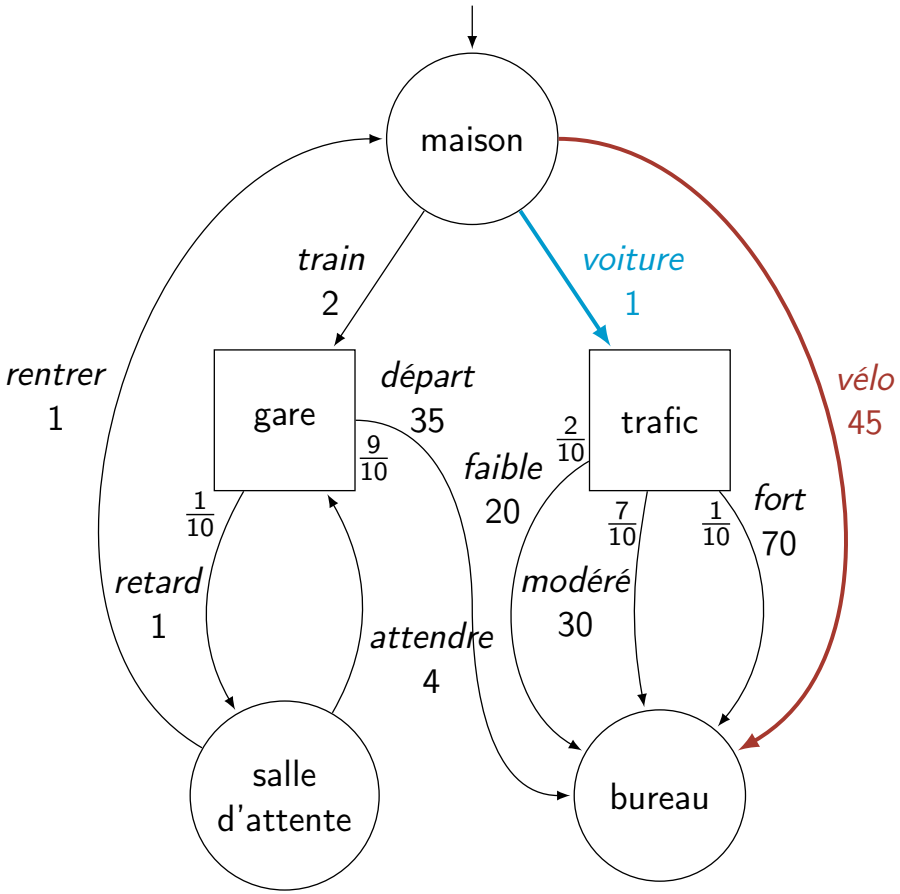
- ▷ Si on ne minimise *que* le temps moyen : voiture.
  - En moyennne $= 33$ minutes.
  - Pire cas $= 71 > 60$ minutes.

Exemple : aller au bureau



- Si on ne minimise *que* le temps moyen : voiture.
  - En moyennne $= 33$ minutes.
  - Pire cas $= 71 > 60$ minutes.
- Si on ne minimise *que* le pire cas : vélo.
  - En moyenne $= 45$ minutes.
  - Pire cas $= 45 < 60$ minutes.

Exemple : aller au bureau



- ▷ **Stratégie APC** : essayer le train jusqu'à 3 retards, puis prendre le vélo.
  - En moyenne ≈ 37.56.
  - Pire cas $= 59 < 60$.
  - Minimise le temps moyen sous la contrainte pire cas.
  - Besoin de **mémoire** !

Quelques contributions

Résultats [BFRR14b]

- **Nouveau modèle** combinant les deux aspects.
- **Algorithme de synthèse** pour jeux de mean-payoff.
- **Algorithme de synthèse** pour jeux de plus court chemin.
- **Caractérisation théorique** complète du problème.

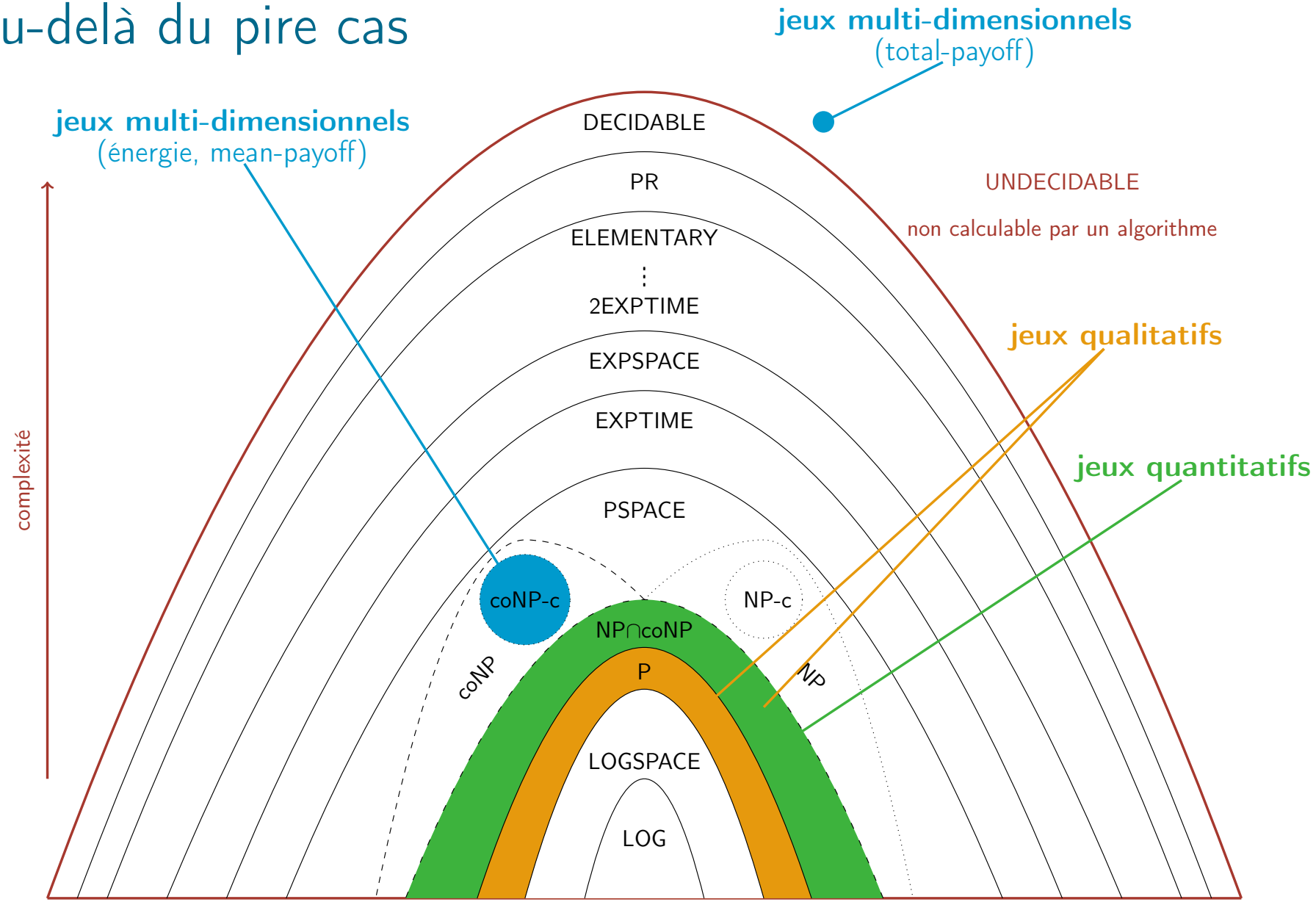
V. Bruyère

E. Filiot

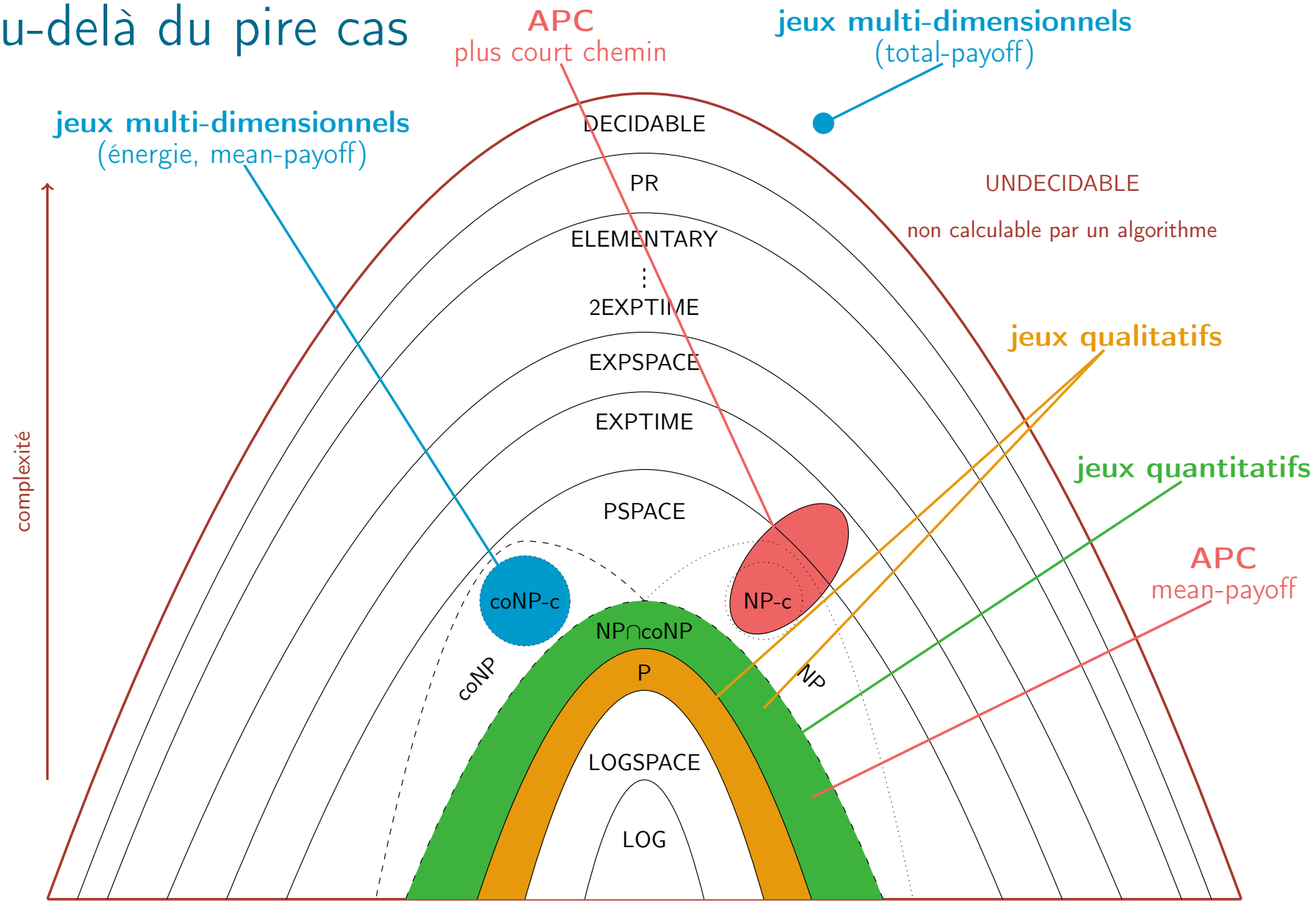
J.-F. Raskin

[BFRR14b] V. Bruyère, E. Filiot, M. Randour, J.-F. Raskin. Meet Your Expectations With Guarantees : Beyond Worst-Case Synthesis in Quantitative Games. STACS'14, LIPIcs 25, 15 pages, S. Dagstuhl, 2014.

Au-delà du pire cas



Au-delà du pire cas



Résumé

- Utilité de la vérification et de la **synthèse**

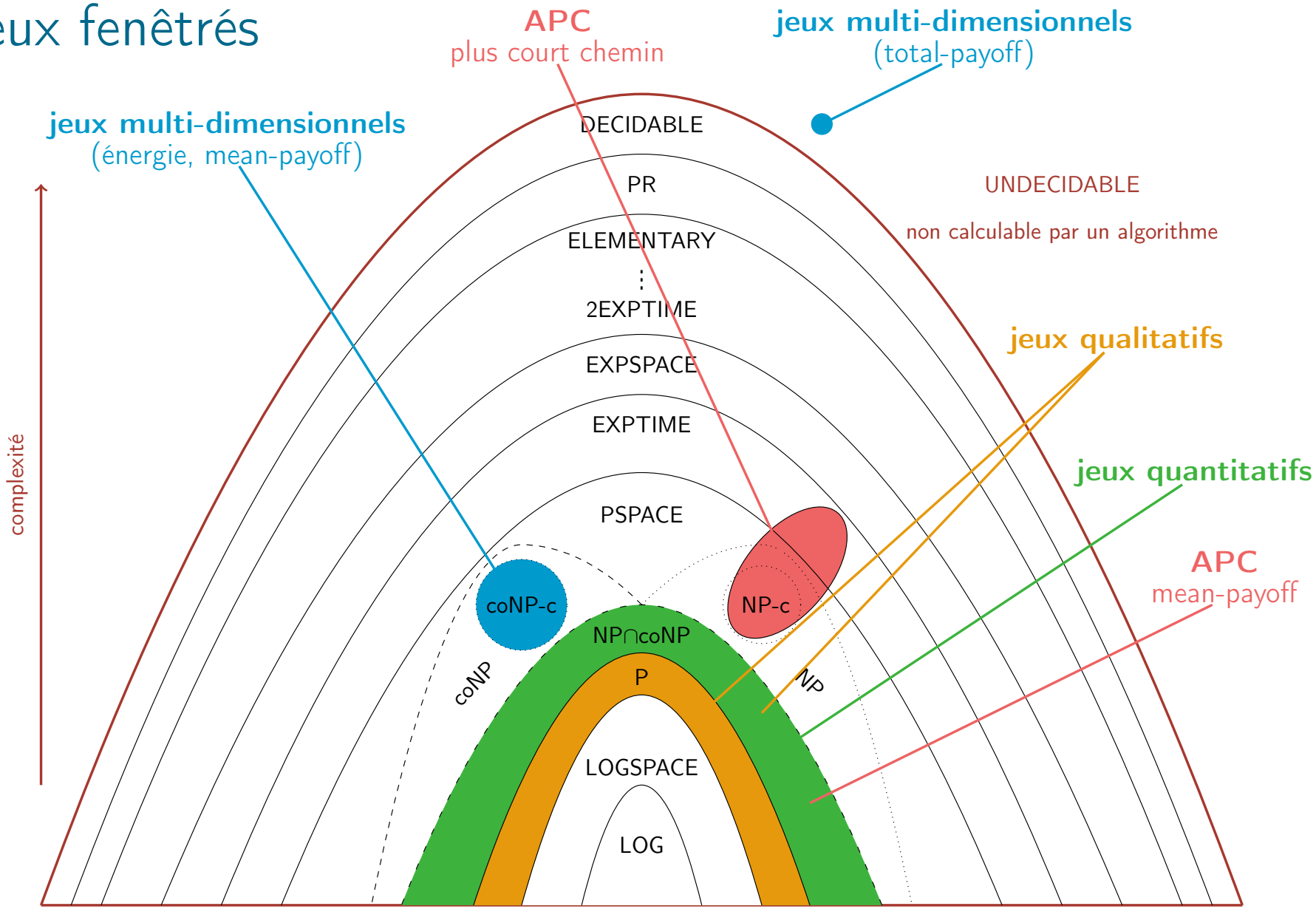
Résumé

- Utilité de la vérification et de la **synthèse**
- Jeux = modèles abstraits
  - qualitatifs
  - quantitatifs
  - **multi-critères**

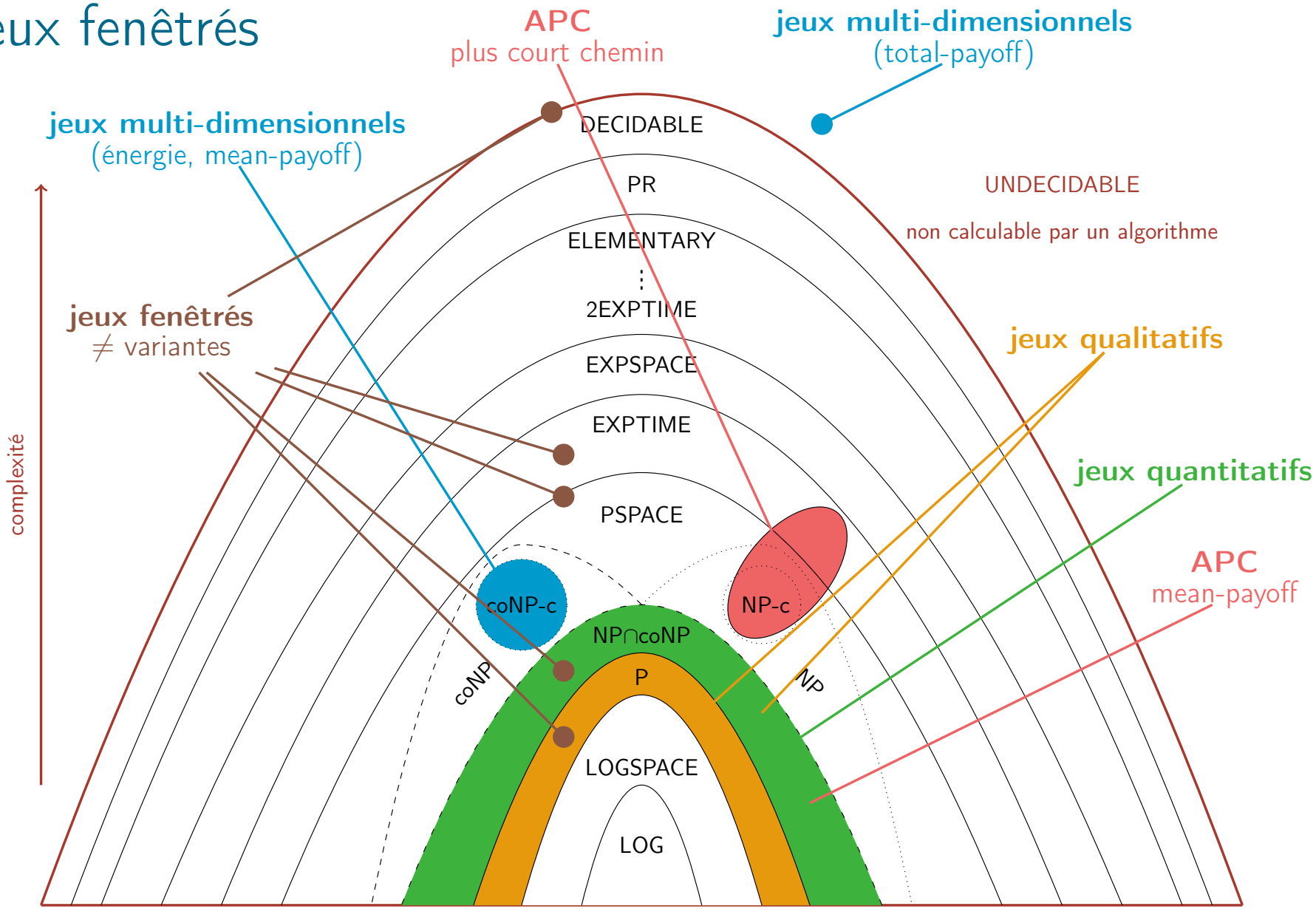
Résumé

- Utilité de la vérification et de la **synthèse**
- Jeux = modèles abstraits
  - qualitatifs
  - quantitatifs
  - **multi-critères**
- **Contributions**
  - jeux multi-dimensionnels
  - au-delà du pire cas
  - jeux *fenêtrés*

Jeux fenêtrés



Jeux fenêtrés



Perspectives

- De nombreuses **questions ouvertes**...
- Besoin de **modèles** de jeux **innovants**
  - ▷ pour capturer la complexité du monde réel,
  - ▷ pour améliorer l'efficacité de nos techniques.

Merci !

A mes promoteurs, Véronique Bruyère et Jean-François Raskin,

à mes co-auteurs, Krishnendu Chatterjee, Laurent Doyen et Emmanuel Filiot,

aux membres du jury,

et bien sûr à vous !

Bibliographie I

Roderick Bloem, Krishnendu Chatterjee, Thomas A. Henzinger, and Barbara Jobstmann.

Better quality in synthesis through quantitative objectives.
In Ahmed Bouajjani and Oded Maler, editors, CAV, volume 5643 of Lecture Notes in Computer Science, pages 140–156. Springer, 2009.

Véronique Bruyère, Emmanuel Filiot, Mickael Randour, and Jean-François Raskin.

Expectations or guarantees ? i want it all ! a crossroad between games and mdps.
In Fabio Mogavero, Aniello Murano, and Moshe Y. Vardi, editors, Proceedings 2nd International Workshop on Strategic Reasoning, Grenoble, France, April 5-6, 2014, volume 146 of Electronic Proceedings in Theoretical Computer Science, pages 1–8. Open Publishing Association, 2014.

Véronique Bruyère, Emmanuel Filiot, Mickael Randour, and Jean-François Raskin.

Meet Your Expectations With Guarantees : Beyond Worst-Case Synthesis in Quantitative Games.
In Ernst W. Mayr and Natacha Portier, editors, 31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), volume 25 of Leibniz International Proceedings in Informatics (LIPIcs), pages 199–213, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Christel Baier and Joost-Pieter Katoen.

Principles of model checking.
MIT Press, 2008.

Julius Richard Büchi.

Weak second-order arithmetic and finite automata.
Mathematical Logic Quarterly, 6(1-6) :66–92, 1960.

Bibliographie II

Arindam Chakrabarti, Luca de Alfaro, Thomas A. Henzinger, and Mariëlle Stoelinga.

Resource interfaces.

In Rajeev Alur and Insup Lee, editors, EMSOFT, volume 2855 of Lecture Notes in Computer Science, pages 117–133. Springer, 2003.

Krishnendu Chatterjee, Laurent Doyen, Mickael Randour, and Jean-François Raskin.

Looking at mean-payoff and total-payoff through windows.

In Dang Van Hung and Mizuhito Ogawa, editors, ATVA, volume 8172 of Lecture Notes in Computer Science, pages 118–132. Springer, 2013.

Edmund M. Clarke and E. Allen Emerson.

Design and synthesis of synchronization skeletons using branching-time temporal logic.

In Dexter Kozen, editor, Logic of Programs, volume 131 of Lecture Notes in Computer Science, pages 52–71. Springer, 1981.

Edmund M. Clarke, Orna Grumberg, and Doron Peled.

Model checking.

MIT Press, 2000.

Alonzo Church.

Applications of recursive arithmetic to the problem of circuit synthesis.

Summaries of the Summer Institute of Symbolic Logic, 1 :3–50, 1957.

Krishnendu Chatterjee, Mickael Randour, and Jean-François Raskin.

Strategy synthesis for multi-dimensional quantitative objectives.

In Maciej Koutny and Irek Ulidowski, editors, CONCUR, volume 7454 of Lecture Notes in Computer Science, pages 115–131. Springer, 2012.

Bibliographie III

Krishnendu Chatterjee, Mickael Randour, and Jean-François Raskin.

Strategy synthesis for multi-dimensional quantitative objectives.
Acta Informatica, 2013.
35 pages.

Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors.

Automata, Logics, and Infinite Games : A Guide to Current Research, volume 2500 of Lecture Notes in Computer Science. Springer, 2002.

Martin J. Osborne and Ariel Rubinstein.

A Course in Game Theory.
MIT Press, 1994.

Amir Pnueli and Roni Rosner.

On the synthesis of a reactive module.
In POPL, pages 179–190. ACM Press, 1989.

Michael O. Rabin.

Decidability of second-order theories and automata on infinite trees.
Transactions of the American Mathematical Society, 141 :1–35, 1969.

Mickael Randour.

Automated synthesis of reliable and efficient systems through game theory : A case study.
In Thomas Gilbert, Markus Kirkilionis, and Gregoire Nicolis, editors, Proceedings of the European Conference on Complex Systems 2012, Springer Proceedings in Complexity, pages 731–738. Springer, 2013.

Bibliographie IV

Peter J. Ramadge and W. Murray Wonham.

Supervisory control of a class of discrete event processes.
SIAM journal on control and optimization, 25(1) :206–230, 1987.

John von Neumann and Oskar Morgenstern.

Theory of Games and Economic Behavior.
Princeton University Press, 1944.

Moshe Y. Vardi and Pierre Wolper.

An automata-theoretic approach to automatic program verification.
In LICS, pages 332–344. IEEE Computer Society, 1986.

Jeux multi-dimensionnels

<table>
<tr><th colspan="2"></th><th>EG</th><th>$\underline{\text{MP}}$</th><th>$\overline{\text{MP}}$</th><th>$\underline{\text{TP}}$</th><th>$\overline{\text{TP}}$</th></tr>
<tr><td rowspan="3">one-dim.</td><td>complexity</td><td colspan="5">NP $\cap$ coNP</td></tr>
<tr><td>$\mathcal{P}_1$ mem.</td><td colspan="5" rowspan="2">pure memoryless</td></tr>
<tr><td>$\mathcal{P}_2$ mem.</td></tr>
<tr><td rowspan="3">k-dim.</td><td>complexity</td><td colspan="2">coNP-c.</td><td>NP $\cap$ coNP</td><td colspan="2">undec.</td></tr>
<tr><td>$\mathcal{P}_1$ mem.</td><td>pure finite</td><td colspan="3">pure infinite</td><td rowspan="2">-</td></tr>
<tr><td>$\mathcal{P}_2$ mem.</td><td colspan="4">pure memoryless</td></tr>
</table>

Jeux multi-dimensionnels

		EG	MP	$\underline{\text{TP}}$	$\overline{\text{TP}}$
one-dim.	complexity	NP $\cap$ coNP			
	$\mathcal{P}_1$ mem.	pure memoryless			
	$\mathcal{P}_2$ mem.				
k-dim.	complexity	coNP-c.		undec.	
	$\mathcal{P}_1$ mem.	pure finite		-	
	$\mathcal{P}_2$ mem.	pure memoryless			

Jeux à fenêtres

	one-dimension			k-dimension		
	complexity	$\mathcal{P}_1$ mem.	$\mathcal{P}_2$ mem.	complexity	$\mathcal{P}_1$ mem.	$\mathcal{P}_2$ mem.
$\underline{\text{MP}}$ / $\overline{\text{MP}}$	NP $\cap$ coNP	mem-less		coNP-c. / NP $\cap$ coNP	infinite	mem-less
$\underline{\text{TP}}$ / $\overline{\text{TP}}$	NP $\cap$ coNP	mem-less		**undec.**	-	-
WMP : fixed polynomial window	**P-c.**	**mem. req.** $\leq$ **linear**($\lvert S\rvert \cdot l_{\max}$)		**PSPACE-h.** **EXP-easy**	**exponential**	
WMP : fixed arbitrary window	**P**($\lvert S\rvert, V, l_{\max}$)			**EXP-c.**		
WMP : bounded window problem	**NP $\cap$ coNP**	**mem-less**	**infinite**	**NPR-h.**	-	-

Au-delà du pire cas

- **Mean-payoff**

	worst-case	expected value	**BWC**
complexity	NP ∩ coNP	P	**NP ∩ coNP**
memory	memoryless	memoryless	**pseudo-polynomial**

- **Plus court chemin**

	worst-case	expected value	**BWC**
complexity	P	P	**pseudo-poly. / NP-hard**
memory	memoryless	memoryless	**pseudo-poly.**