Automated synthesis of reliable and efficient systems through game theory: a case study

Mickael Randour

UMONS - University of Mons

03.09.2012

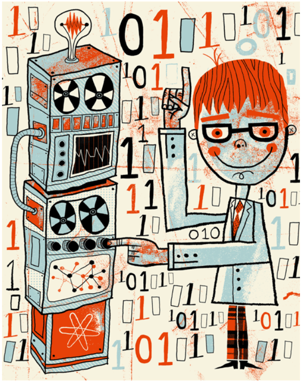
European Conference on Complex Systems

Background

I must confess...

Background

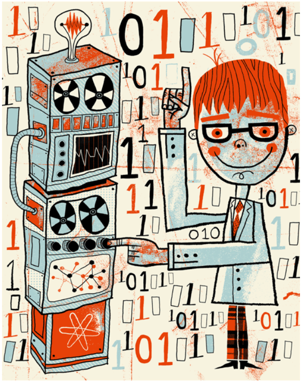
I must confess. . .



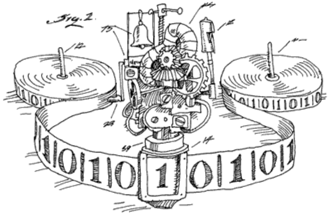
I am a computer scientist.

Background

I must confess. . .



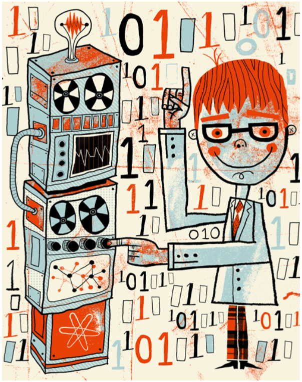
I am a computer scientist.



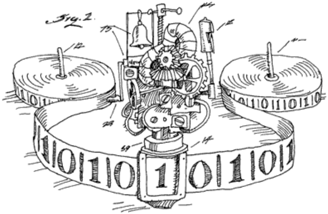
But these are the machines I work with. Focus on **theoretical** computer science.

Background

I must confess. . .



I am a computer scientist.

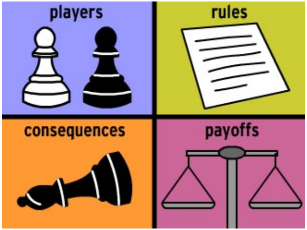


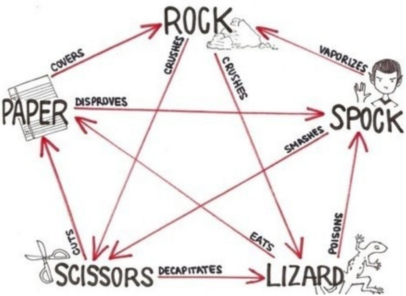
But these are the machines I work with. Focus on **theoretical** computer science.

Turing machine: abstract model of computing device.

Background

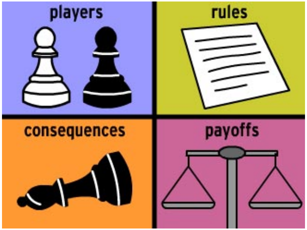
My tools are **games** [VNM44].

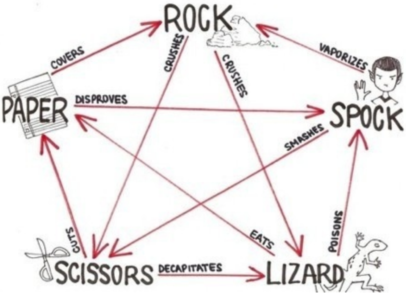




Background

My tools are **games** [VNM44].





Our fields are different. Our games also. Could we still enrich each other's ideas? I certainly hope so!

$\Rightarrow$ high level talk, insight on the problems and concepts.

1 Context

2 Case study

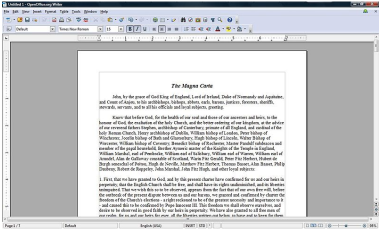
3 Final words

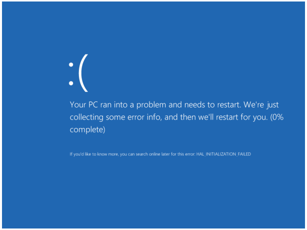
Reactive (computer) systems

- Continuous interaction with the **environment**, must *react* to incoming events.
- Huge, intricate systems $\rightsquigarrow$ bug- and error-prone.

Reactive (computer) systems

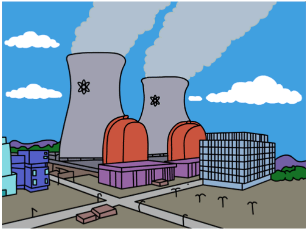
- Continuous interaction with the **environment**, must *react* to incoming events.
- Huge, intricate systems $\rightsquigarrow$ bug- and error-prone.
  - ▷ **Testing** to detect and correct faults.
  - ▷ If there remain faults, we can still issue a *patch* later. . .





Critical systems

- Some systems **do not** tolerate bugs.
  - ▷ Testing is not enough!



- Small flaws can have disastrous consequences!
  - ▷ Therac-25 radiation therapy: several deaths.
  - ▷ Pentium II division unit: $\sim$ 500 million \$.
  - ▷ Ariane 5 explosion (large number conversion).
  - ▷ Mars Climate Orbiter loss (imperial vs. metric).

Formal proof of correctness

- We need mathematical proof that a system will enforce *a correct behavior*, regardless of its environment.
- **Specification**: states what it should do *and* what it should not do.
- Whole systems are too complex: need accurate abstract **models** to work on.
- Two approaches:

Formal proof of correctness

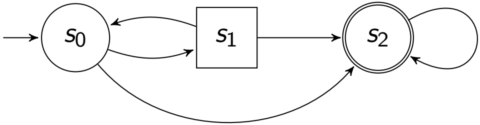
- We need mathematical proof that a system will enforce *a correct behavior*, regardless of its environment.
- **Specification**: states what it should do *and* what it should not do.
- Whole systems are too complex: need accurate abstract **models** to work on.
- Two approaches:
  - ▷ **Verification**: check if an existing system (model) satisfies a given specification, *a posteriori* process [AHK02].

Formal proof of correctness

- We need mathematical proof that a system will enforce *a correct behavior*, regardless of its environment.
- **Specification**: states what it should do *and* what it should not do.
- Whole systems are too complex: need accurate abstract **models** to work on.
- Two approaches:
  - ▷ **Verification**: check if an existing system (model) satisfies a given specification, *a posteriori* process [AHK02].
  - ▷ **Synthesis**: automatically build a correct system *from* the specification, *a priori* process [Chu62, PR89, RW87].

Graph games

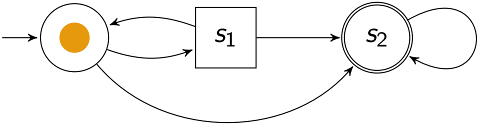
- Model interactions between two players: the system (○) and its adversary, the uncontrollable environment (□).



▷ *states* and *transitions*.

Graph games

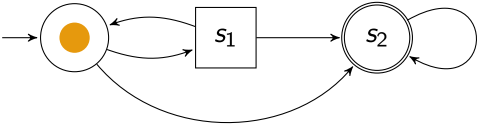
- Model interactions between two players: the system (○) and its adversary, the uncontrollable environment (□).



▷ *Play* begins in initial state: imagine a pebble marking the current state.

Graph games

- Model interactions between two players: the system (○) and its adversary, the uncontrollable environment (□).

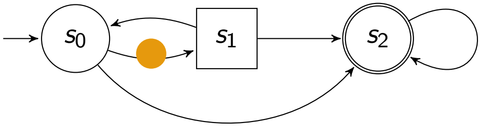


▷ Players take turns: the owner of the state decides where goes the pebble.

▷ Players follow *strategies*: mappings from histories to choices. May be complex! E.g., randomization, memory.

Graph games

- Model interactions between two players: the system (○) and its adversary, the uncontrollable environment (□).

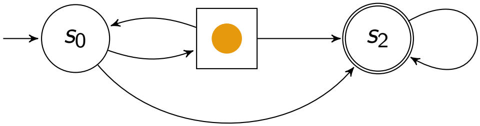


▷ Players take turns: the owner of the state decides where goes the pebble.

▷ Players follow *strategies*: mappings from histories to choices. May be complex! E.g., randomization, memory.

Graph games

- Model interactions between two players: the system (○) and its adversary, the uncontrollable environment (□).

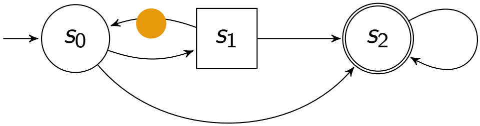


▷ Players take turns: the owner of the state decides where goes the pebble.

▷ Players follow *strategies*: mappings from histories to choices. May be complex! E.g., randomization, memory.

Graph games

- Model interactions between two players: the system (○) and its adversary, the uncontrollable environment (□).

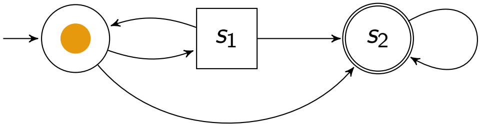


▷ Players take turns: the owner of the state decides where goes the pebble.

▷ Players follow *strategies*: mappings from histories to choices. May be complex! E.g., randomization, memory.

Graph games

- Model interactions between two players: the system (○) and its adversary, the uncontrollable environment (□).

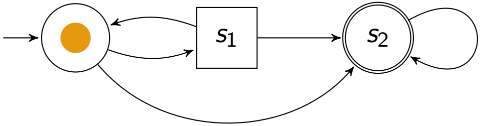


▷ Players take turns: the owner of the state decides where goes the pebble.

▷ Players follow *strategies*: mappings from histories to choices. May be complex! E.g., randomization, memory.

Graph games

- Model interactions between two players: the system (○) and its adversary, the uncontrollable environment (□).

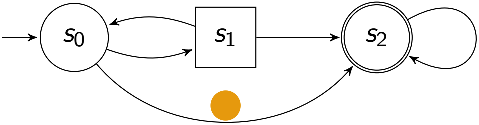


▷ Play continues ad infinitum. Declared *winning* for the system if it *satisfies the specification*. Otherwise, the environment wins. Hence, *zero-sum* games.

▷ E.g., must visit s_2 infinitely often.

Graph games

- Model interactions between two players: the system (○) and its adversary, the uncontrollable environment (□).

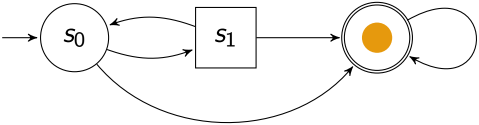


▷ Play continues ad infinitum. Declared *winning* for the system if it *satisfies the specification*. Otherwise, the environment wins. Hence, *zero-sum* games.

▷ E.g., must visit s_2 infinitely often.

Graph games

- Model interactions between two players: the system (○) and its adversary, the uncontrollable environment (□).

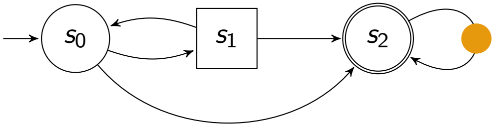


▷ Play continues ad infinitum. Declared *winning* for the system if it *satisfies the specification*. Otherwise, the environment wins. Hence, *zero-sum* games.

▷ E.g., must visit s_2 infinitely often.

Graph games

- Model interactions between two players: the system (○) and its adversary, the uncontrollable environment (□).

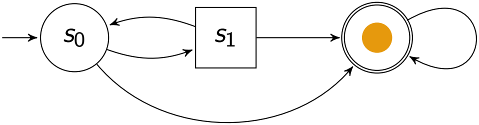


▷ Play continues ad infinitum. Declared *winning* for the system if it *satisfies the specification*. Otherwise, the environment wins. Hence, *zero-sum* games.

▷ E.g., must visit s_2 infinitely often.

Graph games

- Model interactions between two players: the system (○) and its adversary, the uncontrollable environment (□).

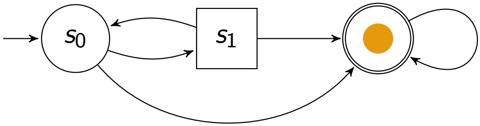


▷ Play continues ad infinitum. Declared *winning* for the system if it *satisfies the specification*. Otherwise, the environment wins. Hence, *zero-sum* games.

▷ E.g., must visit s_2 infinitely often.

Graph games

- Model interactions between two players: the system (○) and its adversary, the uncontrollable environment (□).



▷ A reliable system must win *against any strategy of the environment*.

▷ **Finding a winning strategy for the system = synthesizing a correct controller.**

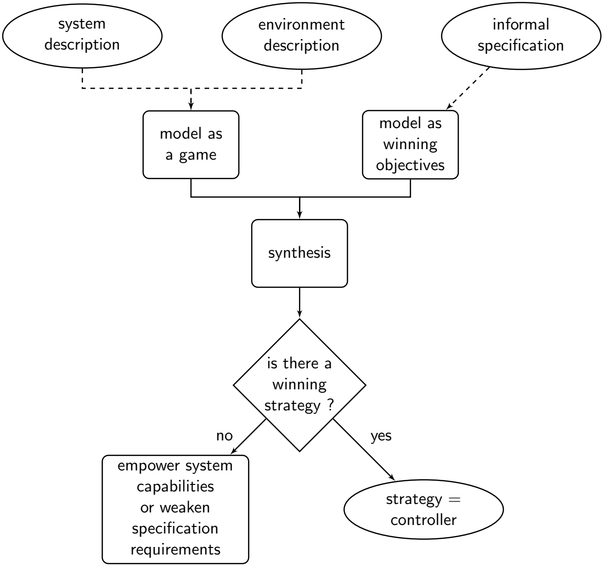
Study of game models: goals

- Study various, powerful classes of games, winning objectives, strategies.
  - ▷ Modeling power vs. tractability.
- Develop efficient, practically useable synthesis algorithms.

Study of game models: goals

- Study various, powerful classes of games, winning objectives, strategies.
  - ▷ Modeling power vs. tractability.
- Develop efficient, practically useable synthesis algorithms.
- Kind of questions:
  - ▷ Can we *decide* if the system can win?
  - ▷ If it can, how complex need its *strategy* be? E.g., does it need memory? How much? Does it need to be randomized?
  - ▷ How complex is it to *build* such a strategy? Time and space complexity?

Synthesis process



Toy example: the automated lawnmower

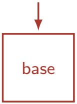
- Goal: synthesize a controller for a robotized lawnmower.

- Illustrates recent results of Chatterjee, Randour and Raskin [CRR12] on the synthesis problem for
  - *qualitative behaviors* (e.g., always eventually granting requests, never reaching a deadlock),
  - along with *multiple quantitative requirements* (e.g., maintaining a bound on the mean response time, never running out of energy).

Modeling as a game

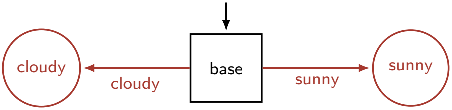
- Model the interactions between the lawnmower and its environment as a game.
- Model the specification to enforce as winning objectives for the lawnmower.

Modeling as a game



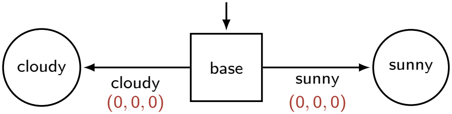
▷ The lawnmower starts the game in its base.

Modeling as a game



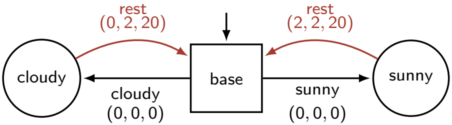
▷ The weather can be cloudy or sunny.

Modeling as a game



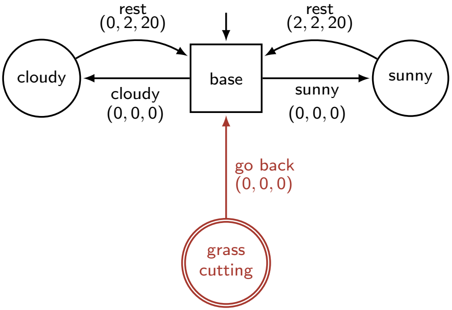
- Electric battery recharged under sunshine thanks to solar panels. Fuel tank filled on the base. Both are unbounded.
- Each action takes time.

Modeling as a game



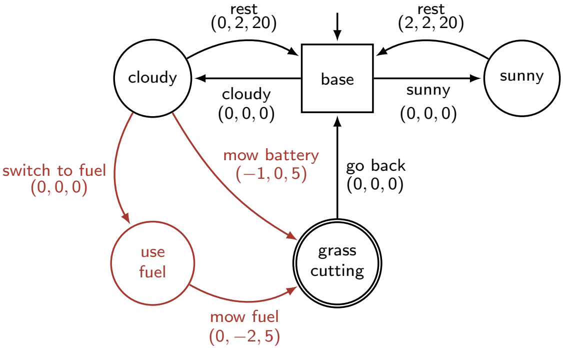
- Recharge battery (2 battery units) only when sunny.
- Refuel (2 fuel units) under both weather conditions.
- Resting takes 20 time units.

Modeling as a game



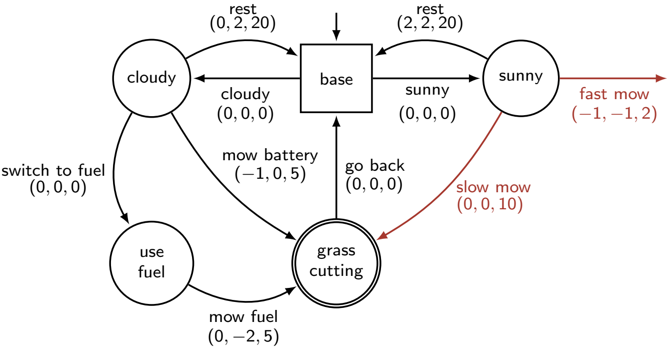
- ▷ No bound on the frequency of grass-cuttings.
- ▷ However, the grass must not grow boundlessly $\rightsquigarrow$ the lawnmower should cut the grass infinitely often.

Modeling as a game



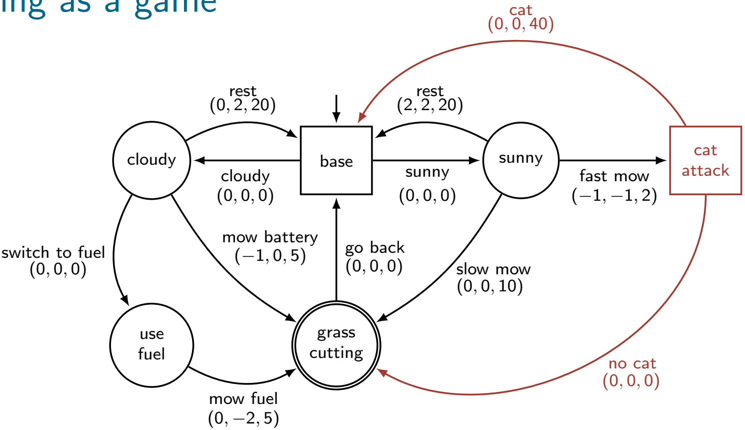
- When cloudy, operate under battery (1 battery unit) or using fuel (2 fuel units).
- Same speed (5 time units).

Modeling as a game



- When sunny, slowly consumes no energy but takes 10 time units.
- Fast consumes both 1 unit of fuel and 1 unit of battery, but only takes 2 time units.

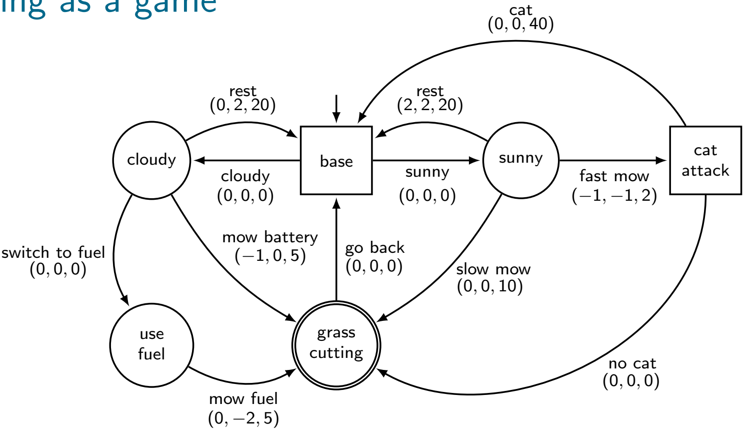
Modeling as a game



▷ Fast makes much noise and may wake up the cat $\rightsquigarrow$ grass-cutting interrupted and 40 time units lost.

▷ The cat does not go out if the weather is bad.

Modeling as a game



▷ What is the objective of the lawnmower, i.e., the specification to enforce?

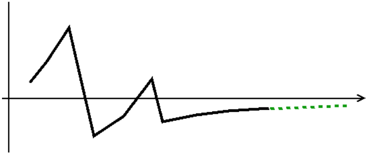
Winning objectives

- *Energy objective*: fuel and battery must never drop below zero.

Winning objectives

- *Energy objective*: fuel and battery must never drop below zero.

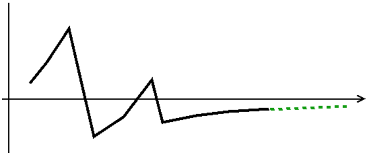
- *Mean-payoff objective*: mean time per action should be less than 10.



Winning objectives

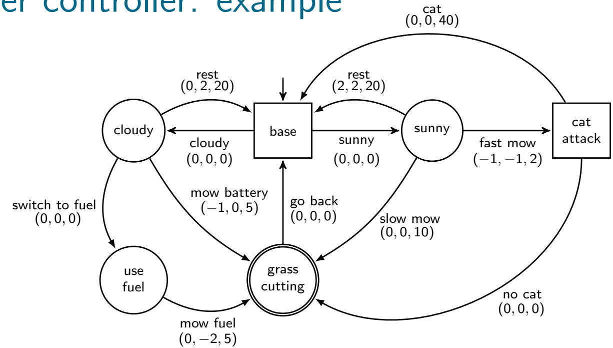
- *Energy objective*: fuel and battery must never drop below zero.

- *Mean-payoff objective*: mean time per action should be less than 10.



- *Infinitely frequent* grass-cutting: infinite visits along a play.

Lawnmower controller: example



Simple controller (needs some memory):

- Start with empty battery and fuel levels.
- If sunny, mow slowly.
- If cloudy,
  - ▷ if battery ≥ 1, mow on battery,
  - ▷ otherwise, if fuel ≥ 2, mow on fuel,
  - ▷ otherwise, rest at the base.

Controller synthesis in a nutshell: 1/2

Result 1 (Induced by [CRR12, Theorem 1]).

Enforcing a specification combining both qualitative and quantitative aspects may require **exponential size** controllers in terms of memory requirements in the worst case.

- ▷ Some systems require *huge* controllers.

Controller synthesis in a nutshell: 2/2

- Sound formal bases and practically efficient algorithms for the automated synthesis of provably safe controllers for reactive systems.

Result 2 (Induced by [CRR12, Theorem 2]).

The synthesis of controllers for systems with qualitative and quantitative requirements, such as the lawnmower, is in EXPTIME.

▷ Deciding *if there exists* a good controller is easier: coNP-complete [CDHR10].

The real world is complex

- Our techniques are only as good as our models.

The real world is complex

- Our techniques are only as good as our models.
- We are always looking for new:
  - ▷ game paradigms (concurrent, *n*-player, etc),
  - ▷ winning objectives (e.g., *quantitative measures*),
  - ▷ applications. . .
  - ▷ . . . and questions!
- Maybe we can exchange some ideas?

Thanks. Questions ?

R. Alur, T.A. Henzinger, and O. Kupferman.
Alternating-time temporal logic.
J. ACM, 49(5):672–713, 2002.

K. Chatterjee, L. Doyen, T.A. Henzinger, and J.-F. Raskin.
Generalized mean-payoff and energy games.
In *Proc. of FSTTCS*, LIPIcs 8, pages 505–516. Schloss Dagstuhl - LZI, 2010.

A. Church.
Logic, arithmetic, and automata.
In *Proceedings of the International Congress of Mathematicians*, pages 23–35. Institut Mittag-Leffler, 1962.

K. Chatterjee, M. Randour, and J.-F. Raskin.
Strategy synthesis for multi-dimensional quantitative objectives.
In *Proc. of CONCUR*, LNCS. Springer, 2012.

To appear. Extended version on CoRR: http://arxiv.org/abs/1201.5073.

A. Pnueli and R. Rosner.
On the synthesis of a reactive module.
In *Proc. of POPL*, pages 179–190, 1989.

M. Randour.
Automated synthesis of reliable and efficient systems through game theory: a case study.
In *Proc. of ECCS*, 2012.
To appear. On CoRR: http://arxiv.org/abs/1204.3283.

P.J. Ramadge and W.M. Wonham.
Supervisory control of a class of discrete-event processes.
SIAM Journal of Control and Optimization, 25(1):206–230, 1987.

J. Von Neumann and O. Morgenstern.
Theory of games and economic behavior.

Princeton University Press, 1944.