Verification of properties on partially observable Petri nets

Federica Adobbati

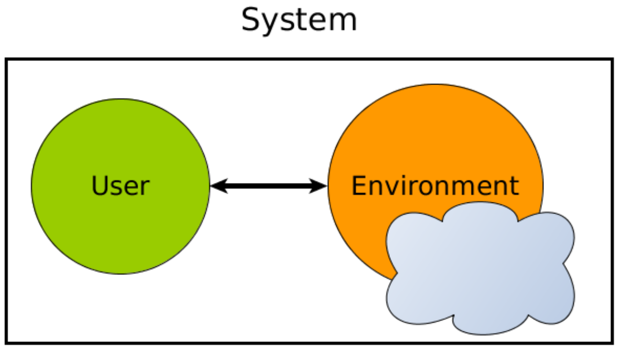
17 June 2021

About me

My name is Federica, I am a PhD student at Milano-Bicocca University

- Bachelor degree in maths
- Master degree in computer science
  - Erasmus in Bielefeld
  - Thesis: Asynchronous games on Petri nets with application to control problems (supervisors: Luca Bernardinello and Lucia Pomello)
- Since November 2019 PhD student in computer science in MC3 lab

Introduction



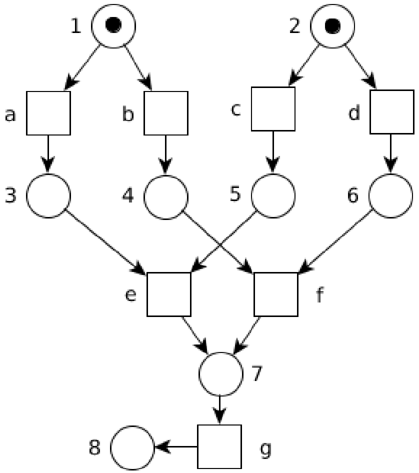
1. What information can the user get through his observations?
2. Is the user able to force some behaviour on the system?

Context

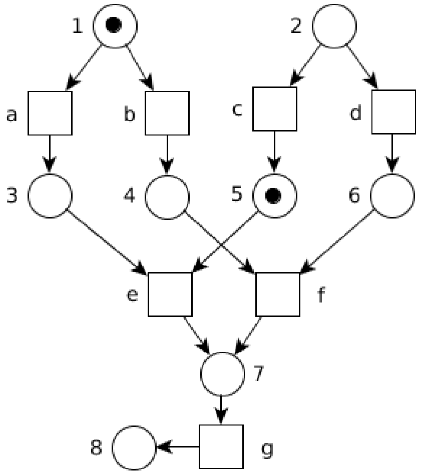
Concurrent systems

- System modelled with a Petri net
- The user knows the structure of the system
- The user observes only a part of the net
- The user has a goal (grabbing information, performing actions...)
- The environment is hostile or indifferent

Modelling language: Petri nets



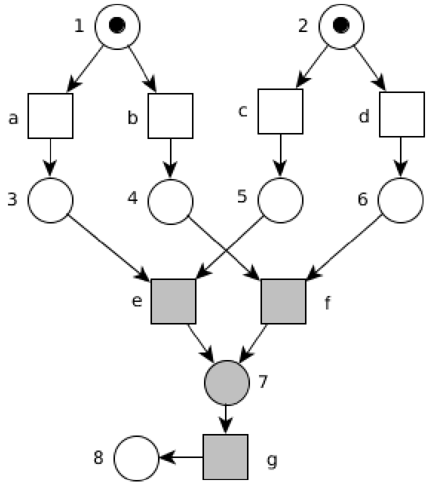
Modelling language: Petri nets



What information can the user get through his observations?

Information flow

○ observable ● unobservable



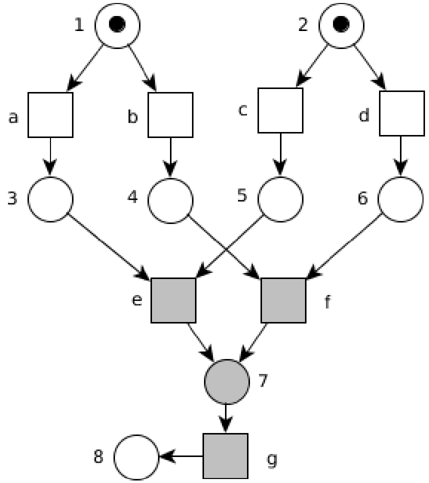
The user interacts only by observing the system

Positive flow

If the user observes *a* and *c*, knows that *g* will eventually fire

Information flow

○ observable ● unobservable

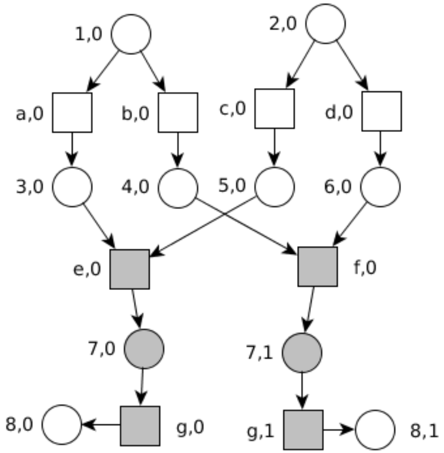


The user interacts only by observing the system

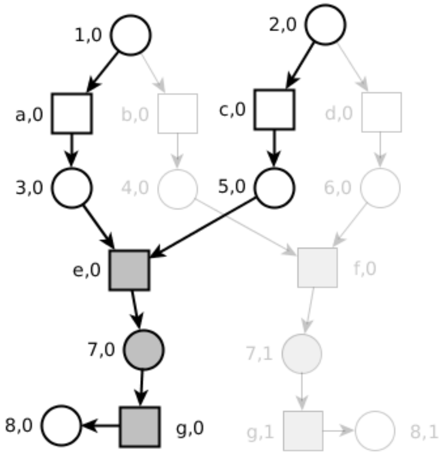
Negative flow

If the user observes *a*, knows that *f* did not and will not fire

Formal definition on the unfolding



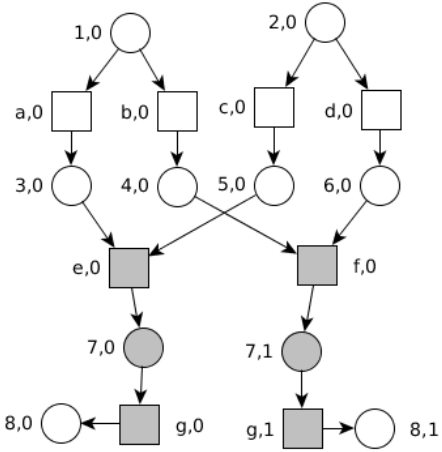
Formal definition on the unfolding



Run

Possible history of what happened in the net

Formal definition on the unfolding



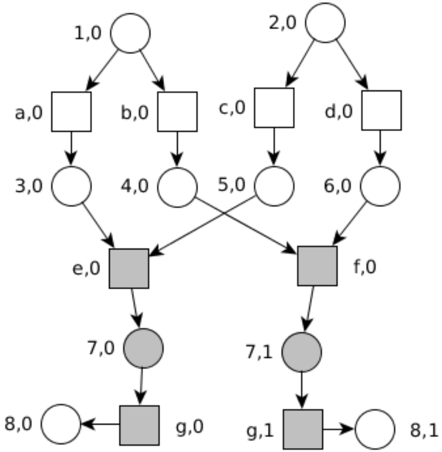
Run

Possible history of what happened in the net

Reveals

$\{a, c\}$ *reveals* g iff in each run with at least an occurrence of a and c, there is also an occurrence of g

Formal definition on the unfolding



Run

Possible history of what happened in the net

Reveals

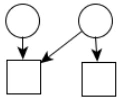
$\{a, c\}$ reveals g iff in each run with at least an occurrence of a and c, there is also an occurrence of g

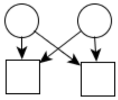
Excludes

a excludes f iff in no run with at least an occurrence of f there is an occurrence of a.

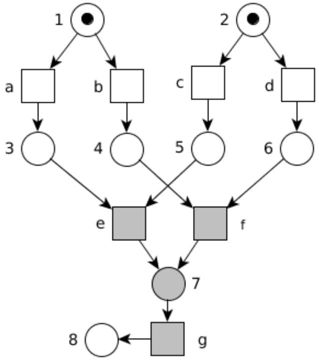
Free choice nets

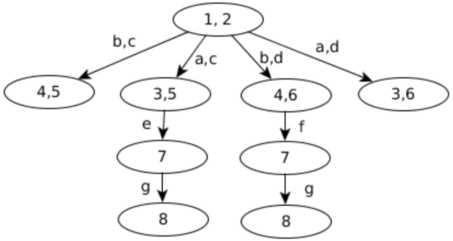
In a free choice net, when two transitions are in conflict share all the preconditions


Non free-choice


Free-choice

The tree of maximal-step computations

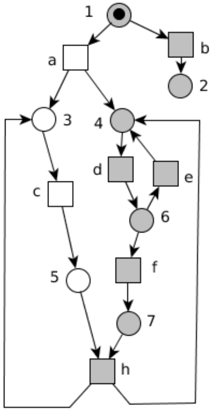


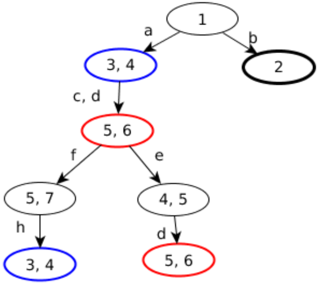


Definition

- Each node is a marking, each branch is labelled with a maximal step enabled in that marking
- A leaf is a deadlock or a repetition in the same run

Another example





c reveals *d*, *c* does not reveal *h*, *a* excludes *b*

Results

Theorem

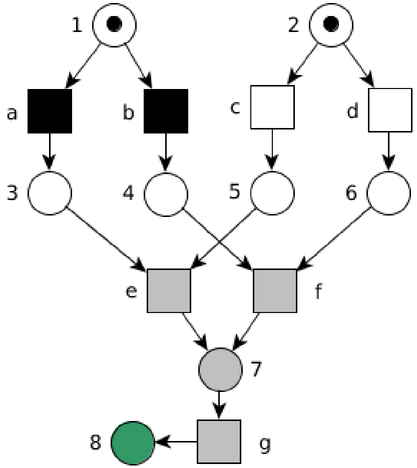
Let $\Sigma = (P, T, F, m_0)$ be a free-choice net, and $a, b \in T$.

1. a reveals b on the unfolding iff in every path of the tree of maximal-step computations with a there is also b.
2. The tree is sufficient to compute all the labels of all the runs in the unfolding. From this sets of labels we can compute excludes and general reveals relations.

Is the user able to force some behavior on the system?

Control of the net

□ uncontrollable ■ controllable ● unobservable ● target



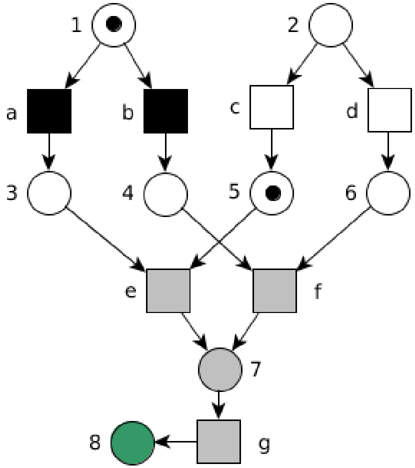
The user interacts by controlling some transitions

Example

Can the user force the system to reach place 8 by controlling only black transitions?

Control of the net

□ uncontrollable ■ controllable ● unobservable ● target



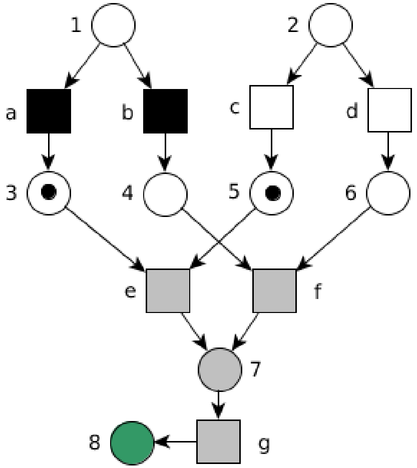
The user interacts by controlling some transitions

Example

Can the user force the system to reach place 8 by controlling only black transitions? Yes, by observing which transition the environment fires

Control of the net

□ uncontrollable ■ controllable ● unobservable ● target



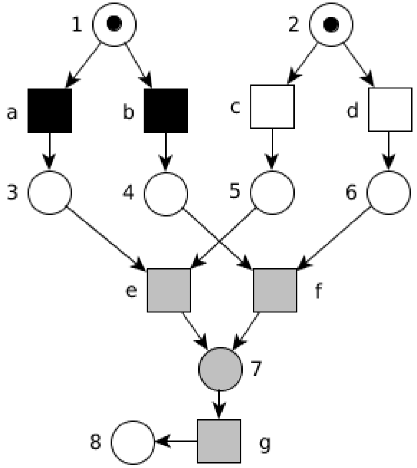
The user interacts by controlling some transitions

Example

Can the user force the system to reach place 8 by controlling only black transitions? Yes, by observing which transition the environment fires

Asynchronous game on Petri nets

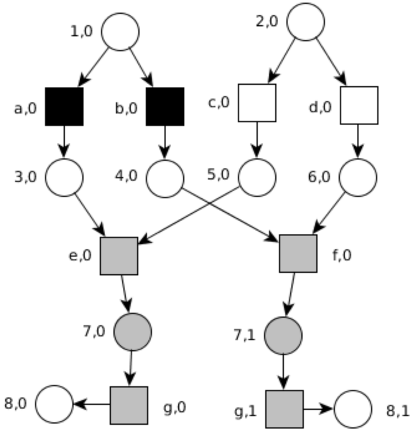
□ uncontrollable ■ controllable ● unobservable



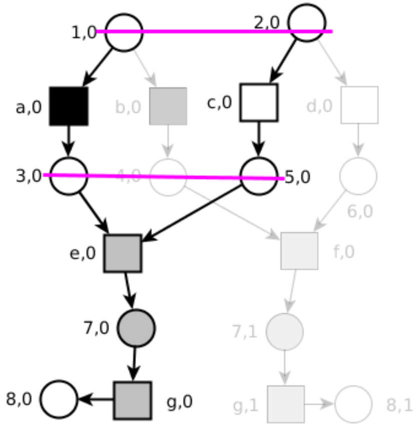
Rules

- Whenever a transition is enabled, its owner can decide to fire it
- The Environment must guarantee the progress of the system

Game on the unfolding



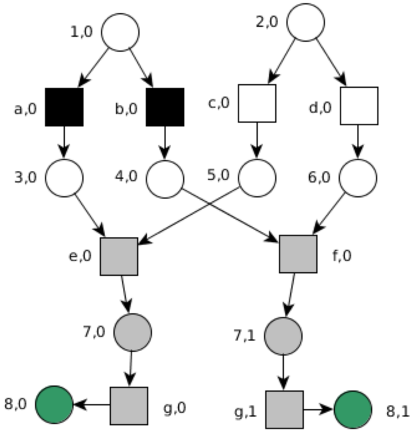
Game on the unfolding



- B-cut: a maximal set of pairwise concurrent places
- Play: a run on the unfolding and an increasing sequence of cuts
- Strategy: $\alpha : obs(\Gamma) \rightarrow 2^{T_u}$

The user has a **winning strategy** iff he wins all the plays that are consistent with it

Game on the unfolding

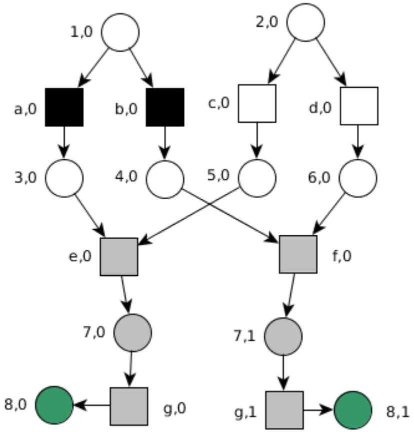


Example

- $\alpha(\{1_0, 5_0\}) = \{a\}$
- $\alpha(\{1_0, 6_0\}) = \{b\}$
- $\alpha(\gamma_o) = \emptyset$ for all other observable cuts

Observation of cuts provides a **memory** to the user

Game on the unfolding



If the strategy is **memoryless** can be defined on *markings*

Example

- $\alpha(\{1,5\}) = \{a\}$
- $\alpha(\{1,6\}) = \{b\}$
- $\alpha(m_o) = \emptyset$ for all other observable markings

Question

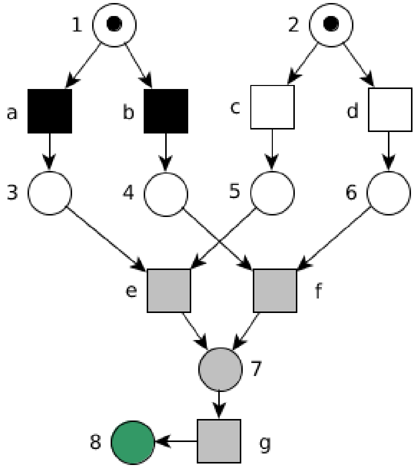
Given a goal for the user and a partially controllable net, how can we decide whether the user has a winning strategy?

Proposed solution

1. Definition of a class of goals
2. Reduction to a game on a finite graph
3. Derivation of a strategy on the Petri net from the strategy on the graph
4. Implementation of the strategy

1. Goals for the users

We can express goals with temporal logics (e.g. LTL)

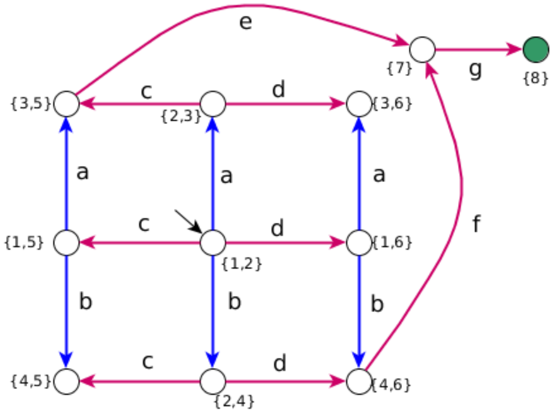


Can the user force the system to reach place 8 by controlling only black transitions?

⇒ Can the user force the validity of $F8$?

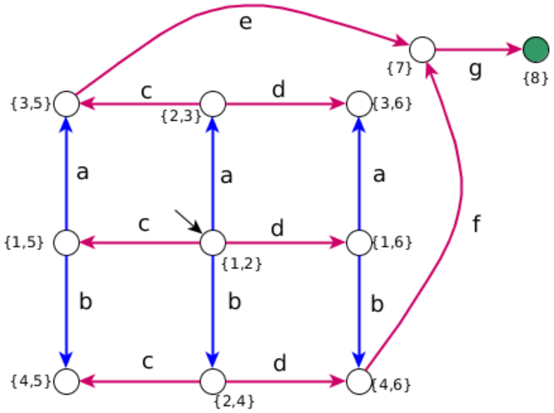
2. Transition systems as Kripke models

⟶ uncontrollable ⟶ controllable ● target



2. Transition systems as Kripke models

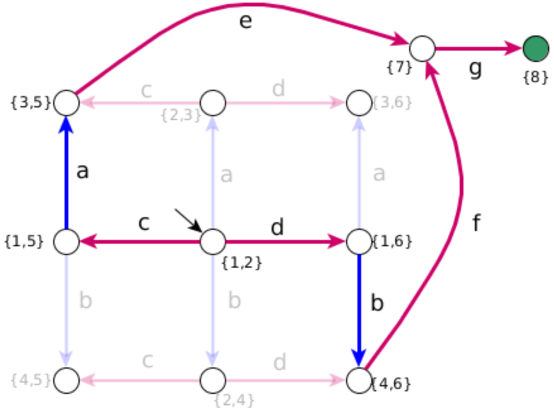
$\longrightarrow$ uncontrollable $\longrightarrow$ controllable ● target



We can verify on this structure the ATL^* formula $\langle\langle user \rangle\rangle F8$

2. Model checking on Kripke models

If the user has no memory, we can remove a subset of controllable arcs, and check whether the resulting structure satisfies the LTL formula



3. Results

If the LTL formulas do not use the 'next' operator, we can construct a game on the Kripke model equivalent to the game on the unfolding

Theorem

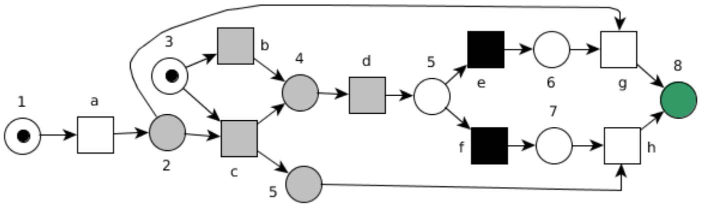
Let G_Σ be the Kripke model derived from the elementary net Σ. The user has a memoryless winning strategy on Σ iff he has a memoryless winning strategy on G_Σ, for all the formulas that do not use the 'next' operator

Advantage: We can use the algorithms for Kripke models to find strategies on Petri nets

Drawback: When we consider memory, this equivalence does not hold because of the different semantics

Example with memory

■ controllable ● unobservable ● target

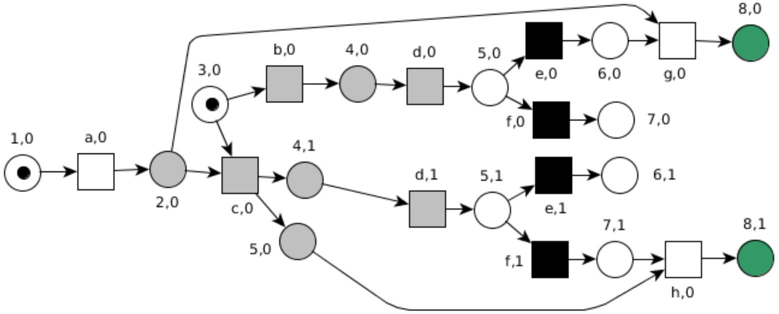


If the user observes place 5, he cannot know which between *e* and *f* he has to choose to reach 8

Memory on the unfolding

■

controllable ● unobservable ● target



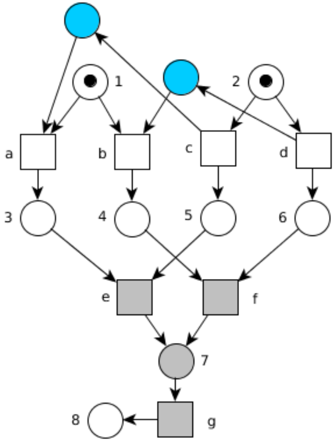
If the user partially observes cuts, there is a winning strategy: $\alpha(5_0) = \{e\}$, $\alpha(5_0) = \{f\}$, $\alpha(\gamma_o) = \emptyset$ otherwise.

Ongoing works

- Development of techniques for finding strategies with causal memory and partial observability
- Development of algorithms working directly on concurrent structures
- Implementable strategies

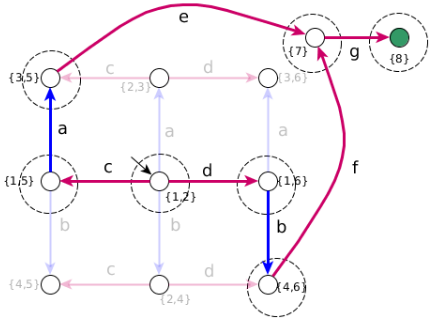
4. Implementable strategies

When there is a winning strategy, we want to know whether we can add places to the net in order to allow only the desired behaviour

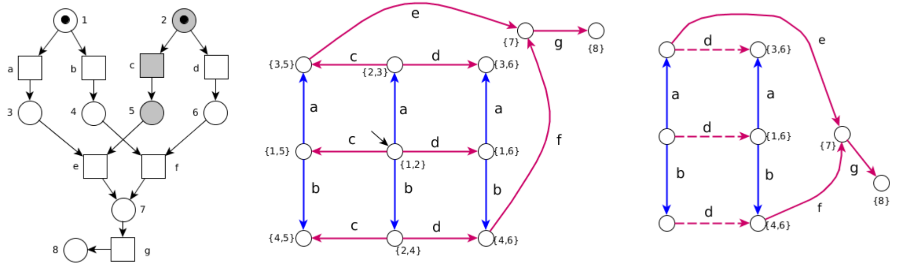


4. Implementable strategies

Conjecture: A strategy is implementable iff the reduced transition system that follows the strategy is synthesizable in a P/T net



Partial observability on the transition system



A sort of modal transition system