Networks of automata with read arcs: a tool for distributed planning

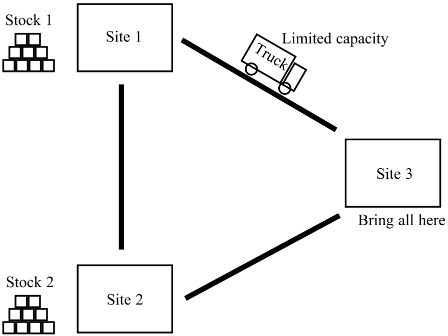
Loïg Jezequel[1], Eric Fabre[2]

IFAC World Congress 2011

[1]ENS Cachan Bretagne
[2]INRIA Rennes Bretagne Atlantique

Outline

Planning and control



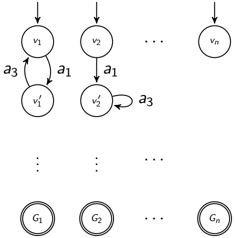
- Variables:
  - truck position
  - truck content
  - content of each site
- Actions:
  - loading truck
  - moving truck
  - unloading truck
- Goal:
  - all items at site 3

Relation with control

Corresponds to a weak form of control problem where everything is observable and controllable.

Formal representation

Several state variables: truck position, stock quantities...



Principle

Each state variable is attached to an automaton:

- states = variable values
- transition = related actions

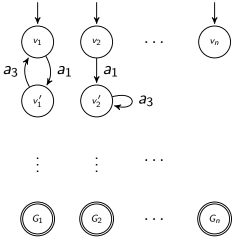
Some actions may synchronously act on several variables, ex: loading the truck.

Objective (centralized point of view)

find a path in the compound system $A_1, \ldots, A_n$ from state $(v_n)_{1 \leq n \leq N}$ to one state in the target $G = \times_n G_n$.

Formal representation

Several state variables: truck position, stock quantities...



Principle

Each state variable is attached to an automaton:

- states = variable values
- transition = related actions

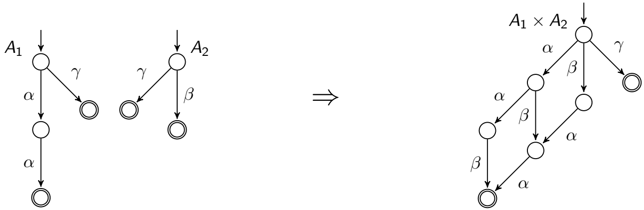
Some actions may synchronously act on several variables, ex: loading the truck.

Objective (distributed point of view)

find a path in each automaton from state v_i to one state in G_i such that these paths are compatible (= coherent use of shared actions).

Centralized approach

1. Compute the product of local automata: $A = A_1 \times \cdots \times A_n$.
2. Search for an accepted word in A.



Global plans

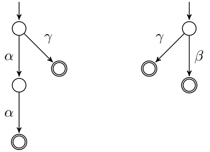
Sequences of actions, here 4 possible accepted words:

1) γ 2) $\alpha\alpha\beta$

3) $\alpha\beta\alpha$ 4) $\beta\alpha\alpha$

Distributed approach

Idea: look for a tuple of local accepted words that coincide on shared actions.



Distributed plans

Partial orders of actions as tuples, here 2 possible tuples:

1) (γ, γ) 2) $(\alpha\alpha, \beta)$

Potential complexity gain

- Reduction of the space of possible plans.
- Coordinated local searches vs. global search.

Existence of reading actions

Concrete example: back to trucks

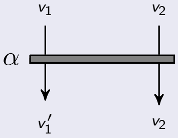
Load_truck and Refuel_truck does not change the position.

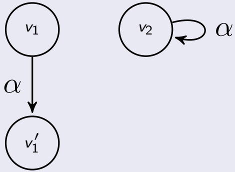
Reading actions

Have preconditions on variables without modifying them.

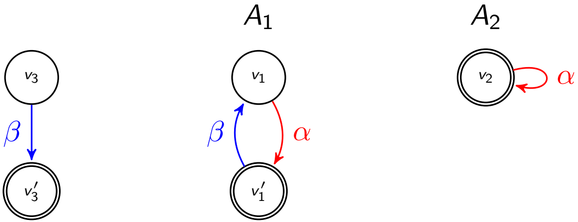
Conventional modeling: reading is an action

ex: α reads V_2 (and modifies V_1).





Drawbacks of conventional modeling



A classical distributed plan: $(\beta, \alpha\beta\alpha, \alpha\alpha)$

Drawback

Readings are counted.

Question: Does A_2 need to know that A_1 uses α two times ?
Answer: No! A_2 can remain idle and only display it state.
What would be a better distributed plan: $(\beta, \alpha\beta\alpha, !v_2)$

Outline

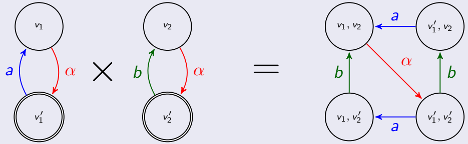
Stating the problem

Variables, their values, and their dynamics = automata

$\mathcal{A} = (S, \Sigma, I, F, T)$

- S=states, Σ=actions, I=initial states, F=final states;
- transition relation: $T \subseteq S \times \Sigma \times S$.

Constraints on variable evolution = synchronous product

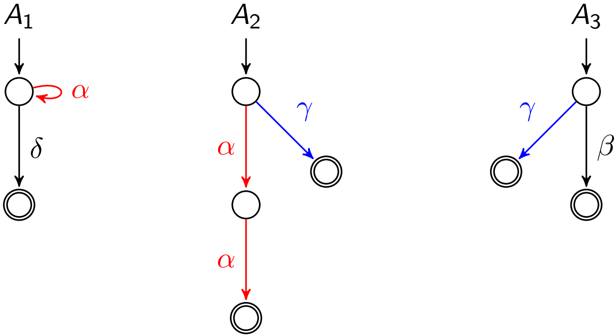


Planning problem ⇒ network of automata

Network: $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$
Interaction graph: edge between $\mathcal{A}_i$ and $\mathcal{A}_j$ iff $\Sigma_i \cap \Sigma_j \neq \emptyset$

Goal

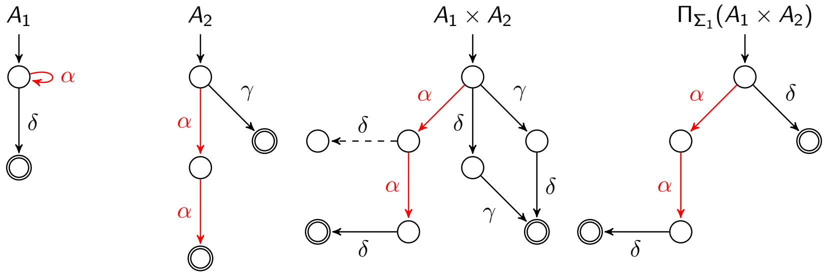
Given $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$, find an accepted path in $\mathcal{A}$ – as a tuple of compatible local paths – without computing $\mathcal{A}$ nor its language.



Solving problems

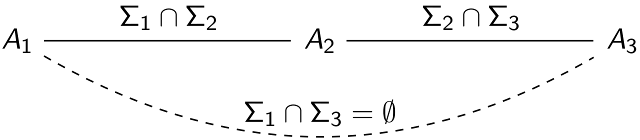
Main remark

The projection (ε-reduction) $\mathcal{A}'_i$ of $\mathcal{A} = A_1 \times \cdots \times A_n$ on Σ_i contains exactly the paths from $\mathcal{A}_i$ that are part of an accepted path in $\mathcal{A}$.



A possible solution

As soon as the interaction graph is a tree one can use message passing algorithm to compute the $\mathcal{A}'_i$ without computing $\mathcal{A}$.



Principle of the MPA

$$\begin{aligned}
\mathcal{A}'_1 &= \Pi_{\Sigma_1}[\mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{A}_3] \\
&= \mathcal{A}_1 \times \Pi_{\Sigma_1 \cap \Sigma_2}[\mathcal{A}_2 \times \mathcal{A}_3] \\
&= \mathcal{A}_1 \times \Pi_{\Sigma_1 \cap \Sigma_2}[\mathcal{A}_2 \times \Pi_{\Sigma_2 \cap \Sigma_3}[\mathcal{A}_3]]
\end{aligned}$$

A possible solution

As soon as the interaction graph is a tree one can use message passing algorithm to compute the $\mathcal{A}'_i$ without computing $\mathcal{A}$.

$$\begin{tikzcd} A_1 \arrow[r, dash, "\Sigma_1 \cap \Sigma_2"] \arrow[rr, dash, dashed, bend right, "\Sigma_1 \cap \Sigma_3 = \emptyset"'] & A_2 \arrow[r, dash, "\Sigma_2 \cap \Sigma_3"] & A_3 \end{tikzcd}$$

Principle of the MPA

$$\begin{tikzcd} A_1 & A_2 \arrow[l, "M_{2,1} = \Pi_{\Sigma_1}(A_2 \times M_{3,2})"'] & A_3 \arrow[l, "M_{3,2} = \Pi_{\Sigma_2}(A_3)"'] \end{tikzcd}$$

$$A'_1 = A_1 \times M_{2,1}$$

Outline

Intuitions on the model: automata

Principle

In a component A_i, each transition:

- can read labels, ex: current state of other automata
- can write (or display) labels, ex: next state of A_i

Sample transition

$$\bigcirc \xrightarrow{\quad r,\alpha,w \quad} \bigcirc$$

- r: readings, a vector with one entry per component,
  - $r(j) = s$ means that s must be read in component A_j,
  - $r(j) = \star$ means nothing special is required from A_j.
- w: writtings,
  - for $j \neq i$, $r(j) = \star$,
  - $r(i) \neq \star$

Intuitions on the model: product

Synchronous product of components: shared actions

$$s_1 \xrightarrow{\binom{o_1}{\star},\ \alpha,\ \binom{o_1'}{\star}} s_1'$$

$$\times$$

$$s_2 \xrightarrow[\binom{o_1}{o_2},\ \alpha,\ \binom{\star}{o_2'}]{} s_2'$$

$$=$$

$$(s_1, s_2) \xrightarrow{\binom{o_1}{o_2},\ \alpha,\ \binom{o_1'}{o_2'}} (s_1', s_2')$$

Intuitions on the model: product

Synchronous product of components: private actions

$$s_1 \xrightarrow{\binom{o_1}{o_2},\, a,\, \binom{o_1'}{\star}} s_1'$$

$$\times$$

$$s_2$$

$$=$$

$$(s_1, s_2) \xrightarrow{\binom{o_1}{o_2},\, a,\, \binom{o_1'}{o_2}} (s_1', s_2)$$

$$s_1 \xrightarrow{\binom{o_1}{\star},\, a,\, \binom{o_1'}{\star}} s_1'$$

$$\times$$

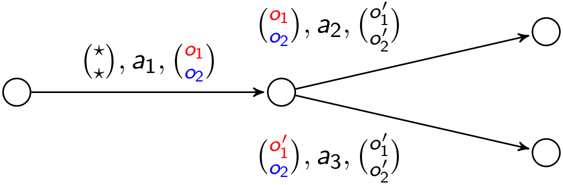
$$s_2$$

$$=$$

$$\begin{tikzcd} (s_1, s_2) \arrow[r, "{\binom{o_1}{o_2},\, a,\, \binom{o_1'}{o_2}}"] \arrow[r, bend right, "{\binom{o_1}{o_2'},\, a,\, \binom{o_1'}{o_2'}}"'] & (s_1', s_2) \end{tikzcd}$$

Intuitions on the model: languages

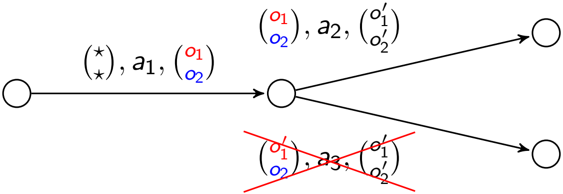
Not all words have to be accepted !



Notion of coherent words: readings and writings must be compatible along the word.

Intuitions on the model: languages

Not all words have to be accepted !



Notion of coherent words: readings and writings must be compatible along the word.

Projection

- Projection is definable in networks of automata with read arcs;
- in the particular networks representing planning problems: $\mathcal{L}(\Pi_{A_i}(A_1 \times A_2)) = \Pi_{A_i}(\mathcal{L}(A_1 \times A_2))$

Theorem

A slightly modified MPA can be used in order to solve planning problems in networks of automata with read arcs, as soon as their interaction graph is a tree.

MPA in presence of read arcs: example

$$A_1 \xleftarrow{M_{2,1} = \Pi_{A_2}(A_2 \times M_{3,2})} A_2 \xleftarrow{M_{3,2} = A_3} A_3$$

$$A_1' = \Pi_{A_1}(A_1 \times M_{2,1})$$

Outline

Results

- Extension to our CDC2009 work;
- reduces the space of possible plans to explore when solving planning problems thanks to:
    - plans represented and computed as partial orders of actions;
    - read arcs mechanism.

Future work

- add costs to automata with read arcs (straightforward) to perform optimal distributed planning;
- perform approximate search rather than exhaustive search of all distributed plans.