

# Distributed Optimal Planning : an Approach by Weighted Automata Calculus

Loïc Jezequel<sup>1</sup>, Éric Fabre<sup>2</sup>

AlgoGT 2011

---

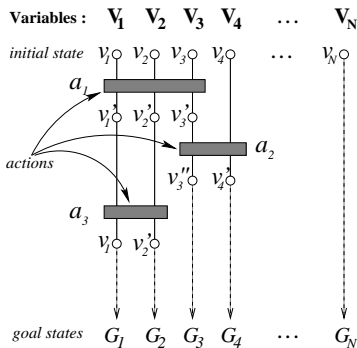
<sup>1</sup>ENS Cachan Bretagne

<sup>2</sup>INRIA Rennes Bretagne Atlantique

# Outline

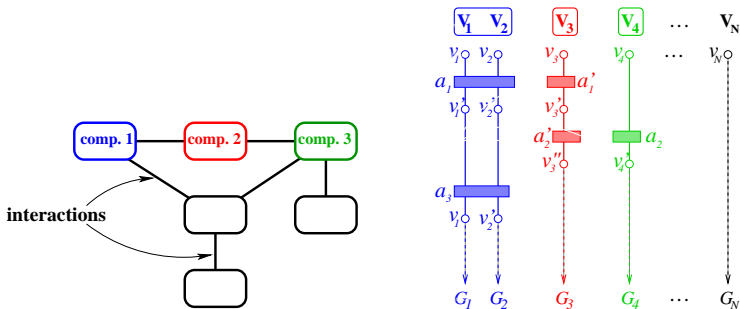
- 1 Motivation
- 2 Problem statement
- 3 Weighted language calculus
- 4 Implementation into WA calculus
- 5 Conclusion
- 6 Perspectives: relations to game theory?

# Motivation: optimal planning



**Objective:** find an action strategy of *minimal cost* to go from system state  $(v_n)_{1 \leq n \leq N}$  to one state in the target  $G = \times_n G_n$

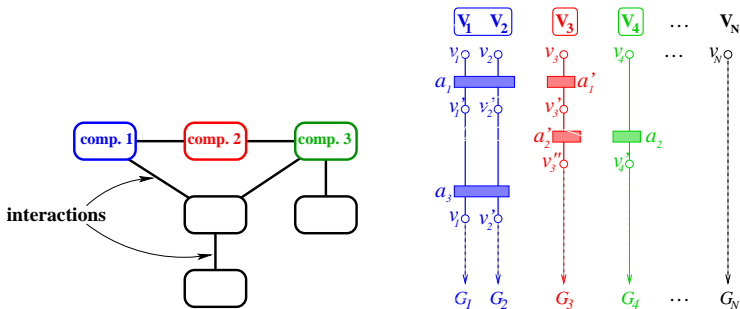
# A *distributed* optimal planning problem



Desired features: a *distributed* resolution

- each component looks for a local plan
- ensure that local plans are *compatible*
- and that their merging yields an *optimal global plan*
- distributed constraint solving + distributed optimization

# A *distributed* optimal planning problem



**Desired features:** a **distributed** resolution

- each component looks for a local plan
- ensure that local plans are **compatible**
- and that their merging yields an **optimal global plan**
- distributed constraint solving + distributed optimization

# Outline

- 1 Motivation
- 2 Problem statement**
- 3 Weighted language calculus
- 4 Implementation into WA calculus
- 5 Conclusion
- 6 Perspectives: relations to game theory?

# Components = weighted automata

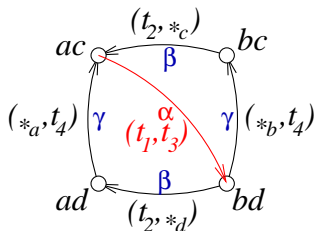
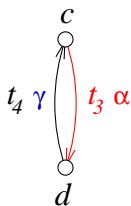
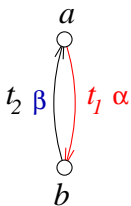
**Weighted automaton:**  $\mathcal{A} = (S, \Sigma, I, F, c, c_i, c_f)$

- $S$ =states,  $\Sigma$ =actions,  $I$ =initial states,  $F$ =final states
- cost function on transitions:  $c : S \times \Sigma \times S \rightarrow \mathbb{K}$   
 $t = (s, \sigma, s') \Rightarrow c(t)$  is the cost for firing  $t$
- **semiring**  $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1}) = (\mathbb{R}^+ \cup \{+\infty\}, \min, +, +\infty, 0)$
- $c_i : I \rightarrow \mathbb{K} \setminus \{\bar{0}\}$  and  $c_f : F \rightarrow \mathbb{K} \setminus \{\bar{0}\}$  are state costs

# Interactions = product of weighted automata

**Synchronous product:**  $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$

- transitions synchronize on common action labels  $\Sigma_1 \cap \Sigma_2$
- costs of synchronized actions are added
- transitions carrying a private action label remain private





# Problem formulation

**Language of a WA (= all weighted plans):**

$$\mathcal{L}(\mathcal{A}) = \sum_{u \in \Sigma^*} \mathcal{L}(\mathcal{A}, u) \cdot u$$

$$\mathcal{L}(\mathcal{A}, u) = \min_{\substack{t_1 t_2 \dots t_n \models \mathcal{A} \\ u = \sigma(t_1 \dots t_n)}} c_i(\bullet t_1) + c(t_1) + \dots + c(t_n) + c_f(t_n^\bullet)$$

**Problem:**

given the network of weighted automata  $\mathcal{A} = \mathcal{A}_1 \times \dots \times \mathcal{A}_N$   
determine the run(s)  $u^*$  of  $\mathcal{A}$  with minimal weight:

$$u^* = \arg \min_{u \in \Sigma^*} \mathcal{L}(\mathcal{A}, u)$$

**Challenge:**

find these runs with a distributed procedure, without  
computing  $\mathcal{A}$  nor  $\mathcal{L}(\mathcal{A})$ ...

# Problem formulation

**Language of a WA (= all weighted plans):**

$$\mathcal{L}(\mathcal{A}) = \sum_{u \in \Sigma^*} \mathcal{L}(\mathcal{A}, u) \cdot u$$

$$\mathcal{L}(\mathcal{A}, u) = \min_{\substack{t_1 t_2 \dots t_n \models \mathcal{A} \\ u = \sigma(t_1 \dots t_n)}} c_i(\bullet t_1) + c(t_1) + \dots + c(t_n) + c_f(t_n^\bullet)$$

**Problem:**

given the network of weighted automata  $\mathcal{A} = \mathcal{A}_1 \times \dots \times \mathcal{A}_N$   
determine the run(s)  $u^*$  of  $\mathcal{A}$  with minimal weight:

$$u^* = \arg \min_{u \in \Sigma^*} \mathcal{L}(\mathcal{A}, u)$$

**Challenge:**

find these runs with a distributed procedure, without  
computing  $\mathcal{A}$  nor  $\mathcal{L}(\mathcal{A})$ ...

# Problem formulation

**Language of a WA (= all weighted plans):**

$$\mathcal{L}(\mathcal{A}) = \sum_{u \in \Sigma^*} \mathcal{L}(\mathcal{A}, u) \cdot u$$

$$\mathcal{L}(\mathcal{A}, u) = \min_{\substack{t_1 t_2 \dots t_n \models \mathcal{A} \\ u = \sigma(t_1 \dots t_n)}} c_i(\bullet t_1) + c(t_1) + \dots + c(t_n) + c_f(t_n^\bullet)$$

**Problem:**

given the network of weighted automata  $\mathcal{A} = \mathcal{A}_1 \times \dots \times \mathcal{A}_N$   
determine the run(s)  $u^*$  of  $\mathcal{A}$  with minimal weight:

$$u^* = \arg \min_{u \in \Sigma^*} \mathcal{L}(\mathcal{A}, u)$$

**Challenge:**

find these runs with a distributed procedure, without  
computing  $\mathcal{A}$  nor  $\mathcal{L}(\mathcal{A})$ ...

# Outline

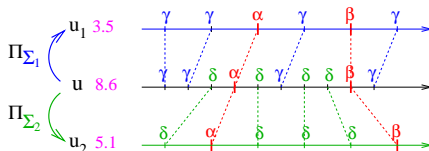
- 1 Motivation
- 2 Problem statement
- 3 Weighted language calculus**
- 4 Implementation into WA calculus
- 5 Conclusion
- 6 Perspectives: relations to game theory?

# Assembling local plans = product of weighted languages

## Product:

- $\mathcal{L}_1, \mathcal{L}_2$  weighted languages on  $\Sigma_1$  and  $\Sigma_2$  resp.
- $\mathcal{L}_1 \times_L \mathcal{L}_2$  is a weighted language on  $\Sigma_1 \cup \Sigma_2$

$$(\mathcal{L}_1 \times_L \mathcal{L}_2)(u) = \mathcal{L}_1(u|_{\Sigma_1}) + \mathcal{L}_2(u|_{\Sigma_2})$$



## Theorem 1

$$\mathcal{L}(\mathcal{A}_1 \times \dots \times \mathcal{A}_N) = \mathcal{L}(\mathcal{A}_1) \times_L \dots \times_L \mathcal{L}(\mathcal{A}_N)$$

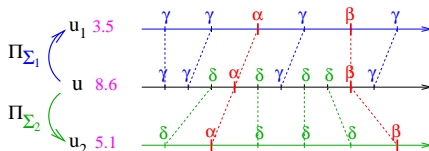
**Consequence:** Global plans have a factorized representation.

# Assembling local plans = product of weighted languages

## Product:

- $\mathcal{L}_1, \mathcal{L}_2$  weighted languages on  $\Sigma_1$  and  $\Sigma_2$  resp.
- $\mathcal{L}_1 \times_L \mathcal{L}_2$  is a weighted language on  $\Sigma_1 \cup \Sigma_2$

$$(\mathcal{L}_1 \times_L \mathcal{L}_2)(u) = \mathcal{L}_1(u|_{\Sigma_1}) + \mathcal{L}_2(u|_{\Sigma_2})$$



## Theorem 1

$$\mathcal{L}(\mathcal{A}_1 \times \dots \times \mathcal{A}_N) = \mathcal{L}(\mathcal{A}_1) \times_L \dots \times_L \mathcal{L}(\mathcal{A}_N)$$

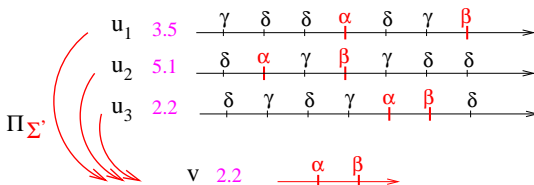
**Consequence:** Global plans have a factorized representation.

# Local view of global plans = projection of a WL

## Projection:

- $\mathcal{L}$  weighted language on  $\Sigma$ , and  $\Sigma' \subseteq \Sigma$
- $\Pi_{\Sigma'}(\mathcal{L})$  is the weighted language on  $\Sigma'$

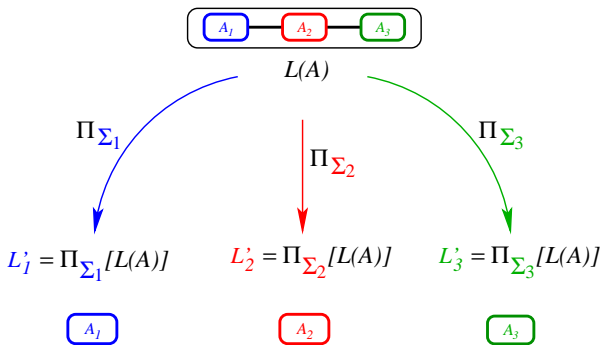
$$[\Pi_{\Sigma'}(\mathcal{L})](v) = \min_{\substack{u \in \Sigma^* \\ u|_{\Sigma'} = v}} \mathcal{L}(u)$$



- $\Rightarrow$  **weight minimization** over words  $u$  with same projection  $v$

# Relating optimal planning to projection

**Objective:** for  $\mathcal{A} = \times_{i \in I} \mathcal{A}_i$  compute the  $\mathcal{L}'_i = \Pi_{\Sigma_i}[\mathcal{L}(\mathcal{A})]$



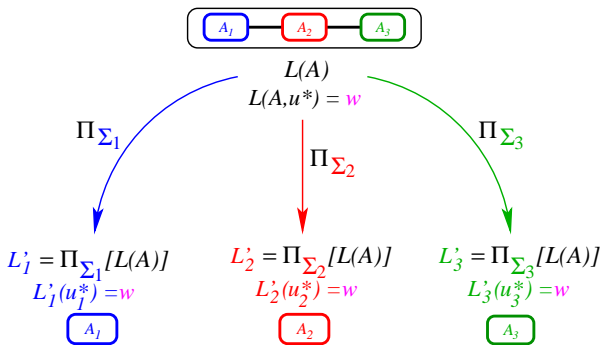
$u^*$  optimal word/plan in  $\mathcal{L}(\mathcal{A})$ , with  $\mathcal{L}(\mathcal{A}, u^*) = w$

$\Rightarrow u_i^* = u_{|\Sigma_i}$  optimal word in  $\mathcal{L}'_i = \Pi_{\Sigma_i}[\mathcal{L}(\mathcal{A})]$ , with  $\mathcal{L}'_i(u_i^*) = w$   
and conversely!



# Relating optimal planning to projection

**Objective:** for  $\mathcal{A} = \times_{i \in I} \mathcal{A}_i$  compute the  $\mathcal{L}'_i = \Pi_{\Sigma_i}[\mathcal{L}(\mathcal{A})]$



$u^*$  optimal word/plan in  $\mathcal{L}(\mathcal{A})$ , with  $\mathcal{L}(\mathcal{A}, u^*) = w$

$\Rightarrow u^*_i = u^*|_{\Sigma_i}$  optimal word in  $\mathcal{L}'_i = \Pi_{\Sigma_i}[\mathcal{L}(\mathcal{A})]$ , with  $\mathcal{L}'_i(u^*_i) = w$   
and conversely!

# The key to distributed optimal planning

## Theorem 2

Let  $\mathcal{L}_1, \mathcal{L}_2$  be weighted languages on  $\Sigma_1, \Sigma_2$  resp.,  
and let  $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma_3 \subseteq \Sigma_1 \cup \Sigma_2$ , then

$$\Pi_{\Sigma_3}(\mathcal{L}_1 \times_L \mathcal{L}_2) = \Pi_{\Sigma_3}(\mathcal{L}_1) \times_L \Pi_{\Sigma_3}(\mathcal{L}_2)$$

- allows us to compute the projections  $\Pi_{\Sigma_i}(\mathcal{L})$  by local computations
- Example: with  $\Sigma_3 = \Sigma_1$ , one has

$$\Pi_{\Sigma_1}(\mathcal{L}_1 \times_L \mathcal{L}_2) = \mathcal{L}_1 \times_L \Pi_{\Sigma_1 \cap \Sigma_2}(\mathcal{L}_2)$$

# The key to distributed optimal planning

## Theorem 2

Let  $\mathcal{L}_1, \mathcal{L}_2$  be weighted languages on  $\Sigma_1, \Sigma_2$  resp.,  
and let  $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma_3 \subseteq \Sigma_1 \cup \Sigma_2$ , then

$$\Pi_{\Sigma_3}(\mathcal{L}_1 \times_L \mathcal{L}_2) = \Pi_{\Sigma_3}(\mathcal{L}_1) \times_L \Pi_{\Sigma_3}(\mathcal{L}_2)$$

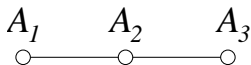
- allows us to compute the projections  $\Pi_{\Sigma_i}(\mathcal{L})$  by local computations
- Example: with  $\Sigma_3 = \Sigma_1$ , one has

$$\Pi_{\Sigma_1}(\mathcal{L}_1 \times_L \mathcal{L}_2) = \mathcal{L}_1 \times_L \Pi_{\Sigma_1 \cap \Sigma_2}(\mathcal{L}_2)$$

## Dynamic programming in space

**Interaction graph of components:** defined by shared actions

Example: assume  $\Sigma_1 \cap \Sigma_3 \subseteq \Sigma_2$



**Important:** we assume the interaction graph is a tree

**Principles of a message passing algorithm:** (example)

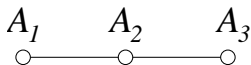
- By Thm 1 :  $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \times_L \mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)$
- By Thm 2:

$$\begin{aligned}
 \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A})] &= \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A}_1) \times_L \mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)] \\
 &= \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A}_1)] \times_L \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)] \\
 &= \mathcal{L}(\mathcal{A}_1) \times_L \Pi_{\Sigma_1} \circ \Pi_{\Sigma_2}[\mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)] \\
 &= \mathcal{L}(\mathcal{A}_1) \times_L \Pi_{\Sigma_1 \cap \Sigma_2}[\mathcal{L}(\mathcal{A}_2) \times_L \Pi_{\Sigma_2 \cap \Sigma_3}[\mathcal{L}(\mathcal{A}_3)]]
 \end{aligned}$$

## Dynamic programming in space

**Interaction graph of components:** defined by shared actions

Example: assume  $\Sigma_1 \cap \Sigma_3 \subseteq \Sigma_2$



**Important:** we assume the interaction graph is a tree

**Principles of a message passing algorithm:** (example)

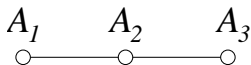
- By Thm 1 :  $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \times_L \mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)$
- By Thm 2:

$$\begin{aligned}
 \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A})] &= \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A}_1) \times_L \mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)] \\
 &= \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A}_1)] \times_L \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)] \\
 &= \mathcal{L}(\mathcal{A}_1) \times_L \Pi_{\Sigma_1} \circ \Pi_{\Sigma_2}[\mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)] \\
 &= \mathcal{L}(\mathcal{A}_1) \times_L \Pi_{\Sigma_1 \cap \Sigma_2}[\mathcal{L}(\mathcal{A}_2) \times_L \Pi_{\Sigma_2 \cap \Sigma_3}[\mathcal{L}(\mathcal{A}_3)]]
 \end{aligned}$$

## Dynamic programming in space

**Interaction graph of components:** defined by shared actions

Example: assume  $\Sigma_1 \cap \Sigma_3 \subseteq \Sigma_2$



**Important:** we assume the interaction graph is a tree

**Principles of a message passing algorithm:** (example)

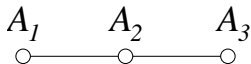
- By Thm 1 :  $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \times_L \mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)$
- By Thm 2:  $(\Sigma_2 \cup \Sigma_3) \cap \Sigma_1 \subseteq \Sigma_1$

$$\begin{aligned}
 \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A})] &= \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A}_1) \times_L \mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)] \\
 &= \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A}_1)] \times_L \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)] \\
 &= \mathcal{L}(\mathcal{A}_1) \times_L \Pi_{\Sigma_1} \circ \Pi_{\Sigma_2}[\mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)] \\
 &= \mathcal{L}(\mathcal{A}_1) \times_L \Pi_{\Sigma_1 \cap \Sigma_2}[\mathcal{L}(\mathcal{A}_2) \times_L \Pi_{\Sigma_2 \cap \Sigma_3}[\mathcal{L}(\mathcal{A}_3)]]
 \end{aligned}$$

## Dynamic programming in space

**Interaction graph of components:** defined by shared actions

Example: assume  $\Sigma_1 \cap \Sigma_3 \subseteq \Sigma_2$



**Important:** we assume the interaction graph is a tree

**Principles of a message passing algorithm:** (example)

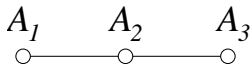
- By Thm 1 :  $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \times_L \mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)$
- By Thm 2:  $(\Sigma_2 \cup \Sigma_3) \cap \Sigma_1 = \Sigma_2 \cap \Sigma_1$

$$\begin{aligned}
 \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A})] &= \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A}_1) \times_L \mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)] \\
 &= \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A}_1)] \times_L \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)] \\
 &= \mathcal{L}(\mathcal{A}_1) \times_L \Pi_{\Sigma_1} \circ \Pi_{\Sigma_2}[\mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)] \\
 &= \mathcal{L}(\mathcal{A}_1) \times_L \Pi_{\Sigma_1 \cap \Sigma_2}[\mathcal{L}(\mathcal{A}_2) \times_L \Pi_{\Sigma_2 \cap \Sigma_3}[\mathcal{L}(\mathcal{A}_3)]]
 \end{aligned}$$

## Dynamic programming in space

**Interaction graph of components:** defined by shared actions

Example: assume  $\Sigma_1 \cap \Sigma_3 \subseteq \Sigma_2$



**Important:** we assume the interaction graph is a tree

**Principles of a message passing algorithm:** (example)

- By Thm 1 :  $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \times_L \mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)$
- By Thm 2:

$$\begin{aligned}
 \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A})] &= \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A}_1) \times_L \mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)] \\
 &= \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A}_1)] \times_L \Pi_{\Sigma_1}[\mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)] \\
 &= \mathcal{L}(\mathcal{A}_1) \times_L \Pi_{\Sigma_1} \circ \Pi_{\Sigma_2}[\mathcal{L}(\mathcal{A}_2) \times_L \mathcal{L}(\mathcal{A}_3)] \\
 &= \mathcal{L}(\mathcal{A}_1) \times_L \Pi_{\Sigma_1 \cap \Sigma_2}[\mathcal{L}(\mathcal{A}_2) \times_L \Pi_{\Sigma_2 \cap \Sigma_3}[\mathcal{L}(\mathcal{A}_3)]]
 \end{aligned}$$



# Outline

- 1 Motivation
- 2 Problem statement
- 3 Weighted language calculus
- 4 Implementation into WA calculus**
- 5 Conclusion
- 6 Perspectives: relations to game theory?

# Implementation is needed and possible

Languages are potentially infinite objects:  
languages are not usable in practice.

## Solution:

- work directly with automata;
- projection:  $\epsilon$ -reduction (+ determinization and minimization);
- product: synchronous product of weighted automata.

# Implementation is needed and possible

Languages are potentially infinite objects:  
languages are not usable in practice.

## Solution:

- work directly with automata;
- projection:  $\epsilon$ -reduction (+ determinization and minimization);
- product: synchronous product of weighted automata.

# Outline

- 1 Motivation
- 2 Problem statement
- 3 Weighted language calculus
- 4 Implementation into WA calculus
- 5 Conclusion**
- 6 Perspectives: relations to game theory?

# Conclusion

## Main features:

- unsupervised distributed search of an optimal plan
- all possible/optimal plans are computed
- global plans are computed as tuples of partially synchronized sequences, *i.e.* as *partial orders* of actions

# Outline

- 1 Motivation
- 2 Problem statement
- 3 Weighted language calculus
- 4 Implementation into WA calculus
- 5 Conclusion
- 6 Perspectives: relations to game theory?**

# Planning as games: why?

## Improve fairness between agents:

- optimizing the sum may penalize some agent;
- is it possible to be fair?

## Reduce communications:

- currently messages are potentially very large (contain all plans);
- in a truly distributed setting they may be numerous;
- is it possible to reduce it?

Game theory may help in solving these issues...

# Planning as games: why?

## Improve fairness between agents:

- optimizing the sum may penalize some agent;
- is it possible to be fair?

## Reduce communications:

- currently messages are potentially very large (contain all plans);
- in a truly distributed setting they may be numerous;
- is it possible to reduce it?

Game theory may help in solving these issues...



# Planning as games: why?

## Improve fairness between agents:

- optimizing the sum may penalize some agent;
- is it possible to be fair?

## Reduce communications:

- currently messages are potentially very large (contain all plans);
- in a truly distributed setting they may be numerous;
- is it possible to reduce it?

Game theory may help in solving these issues. . .

# Planning as games: how?

## "Static" games:

- use for fairness? look for equilibria?
- complete or incomplete information?

## An example:

- one player per automaton;
- a local plan in automaton  $A_i$  is a strategy for player  $i$ ;
- each player has to maximize her payoff (defined below).

$p_i$  is the strategy/local plan chosen by player  $i$ , and  $p$  is any global plan corresponding to all players choice (if exists).

Different possible payoffs:

- payoff for any player =  $1/c(p)$  (or 0 if  $p$  does not exist);
- payoff for player  $i$  =  $1/c(p_i)$  (or 0 if  $p$  does not exist).

# Planning as games: how?

## "Static" games:

- use for fairness? look for equilibria?
- complete or incomplete information?

## An example:

- one player per automaton;
- a local plan in automaton  $A_i$  is a strategy for player  $i$ ;
- each player has to maximize her payoff (defined below).

$p_i$  is the strategy/local plan chosen by player  $i$ , and  $p$  is any global plan corresponding to all players choice (if exists).

Different possible payoffs:

- payoff for any player =  $1/c(p)$  (or 0 if  $p$  does not exist);
- payoff for player  $i$  =  $1/c(p_i)$  (or 0 if  $p$  does not exist).

# Planning as games: how?

## "Dynamic" games:

- dynamic construction of local plans;
- all agents have to agree on a synchronization word;
- to deal with message size/quantity?

