Distributed Reachability Test in Large Distributed Systems

Loïg JEZEQUEL
Supervisor: Eric Fabre

M2R

Monday, the 2^{nd} of February 2009

Introduction : planning problem

Planning problem

- From an initial state, reach a goal state. . .with allowed actions
- Useful in many domains :
  - Artificial intelligence (actions of a robot)
  - Industrial production (scheduling of tasks)

Three levels of difficulty

1. Is there a solution ? (Reachability test)
2. Give a solution.
3. What is the best solution ?

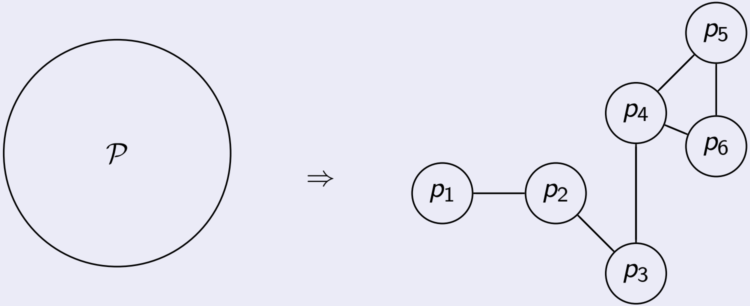
Solutions

- Sequence of actions
- Partial order of actions

Modular planning

Factored planning

Exploits the weak dependencies between variables.



Challenges

- Find the factors (not explored here)
- Find local plans that combine into a global plan

Outline

Automated planning

Automaton

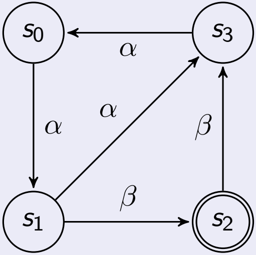
$\mathcal{A} = (S, T, \rightarrow, s_0, \lambda, \Lambda, F)$

Planning

Find a path which reaches the goal.

Example

- $S = \{s_0, s_1, s_2, s_3\}$
- $T = \{t_1, t_2, t_3, t_4, t_5\}$
- $\Lambda = \{\alpha, \beta\}$
- $\lambda(t_1) = \lambda(t_3) = \lambda(t_5) = \alpha$
  $\lambda(t_2) = \lambda(t_4) = \beta$
- $F = \{s_2\}$

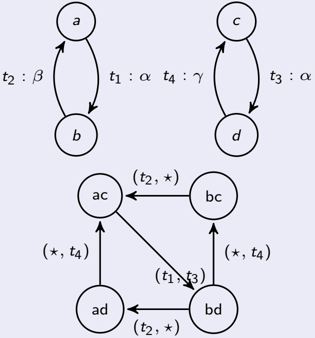


Modular planning

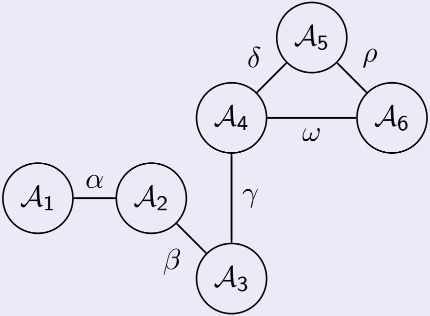
Planning problem

- Network of automata
- Composition by synchronous product
- Local/global plans

Synchronous product

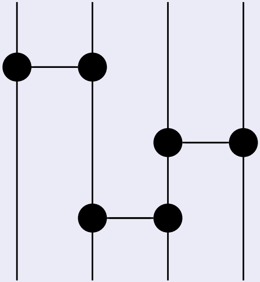


Network of automata



A solution

Idea



Constraint solving

- Landmarks
- Filling

Discussion

- Search for 1 solution
- No backtracking
- No optimization
- Existence of a plan ?

Our approach

Search for all the plans !

Outline

Language theory

From automata to languages

- A language associated to each automaton (in classical manner) : $L(\mathcal{A})$
- $L(\mathcal{A})$ is the set of all local plans for $\mathcal{A}$
- Synchronous product : $L(\mathcal{A}_1 \|_a \mathcal{A}_2) = L(\mathcal{A}_1) \|_l L(\mathcal{A}_2)$

Modular planning problem as languages

A set of languages $\{L_i \subseteq \Sigma_i^* | i \in I\}$ indexed by I

Solutions

- All global plans : $L = L_1 \| \ldots \| L_N$ (very big)
- Another view : $L = E_1 \| \ldots \| E_N$, where $E_i = P_{I,\{i\}}(L)$

Important notions

- Global consistency
- Local consistency

Global Consistency

Definition

$\mathcal{E} = \{E_i \subseteq L_i | i \in I\}$ is globally consistent with respect to I if, $\forall i \in I$, $E_i = P_{I,\{i\}}(\|_{j \in I} E_j)$.

Intuition

For each E_i, knowing all E_j ($j \neq i$) can not help to reduce E_i.

Relation with planning (supremal global support)

A maximal set $\mathcal{E}$ globally consistent with respect to a set of index I gives all the solutions to the global planning problem.

Algorithm for global consistency

How to construct supremal global support ?

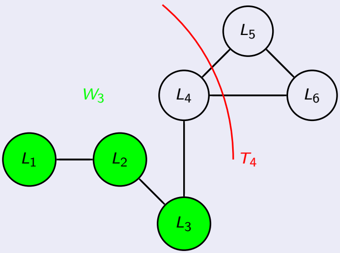
By computing $L = \|_{i \in I} L_i$ and then $E_i = P_{I,\{i\}}(L)$...this is not efficient.

Idea of an algorithm

Alternation of projections and synchronous product, adding components one by one.

Algorithm

1. $T_0 \leftarrow \Sigma_1,\ W_0 \leftarrow \Sigma_1^*$
2. **for** $k = 1$ **to** $n-1$ **do**
    1. $J_k \leftarrow \{1, \ldots, k\}$
    2. $T_k \leftarrow \Sigma_{J_k} \cap \Sigma_{I \setminus J_k}$
    3. $W_k \leftarrow P_{T_{k-1} \cup \Sigma_k, T_k}(W_{k-1} \| L_k)$
3. $E_n \leftarrow W_{n-1} \| L_n$



Local consistency

Definition

$\mathcal{E} = \{E_i \subseteq L_i | i \in I\}$ is locally consistent with respect to I if, $\forall i, j \in I$, $P_{\{i\},\{j\}}(E_i) = P_{\{j\},\{i\}}(E_j)$.

Intuition

For E_i, knowing E_j ($j \neq i$) for all neighbors can not help to reduce E_i.

Relation with global consistency

$$\Sigma_1 = \{\alpha, \beta\}, \Sigma_2 = \{\alpha, \gamma\}, \Sigma_3 = \{\beta, \gamma\}$$
$$L_1 = \{\alpha\beta\}, L_2 = \{\gamma\alpha\}, L_3 = \{\beta\gamma\}$$

local :

$P_{\{1\},\{2\}}(L_1) = P_{\{2\},\{1\}}(L_2)$
$P_{\{1\},\{3\}}(L_1) = P_{\{3\},\{1\}}(L_3)$
$P_{\{2\},\{3\}}(L_2) = P_{\{3\},\{2\}}(L_3)$

global :

$L_1 \| L_2 \| L_3 = \emptyset$

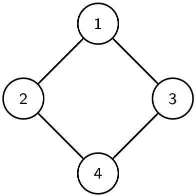
Local consistency

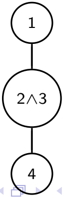
Interests of supremal local support

- An over-approximation of supremal global support
- Simpler to compute than supremal global support
- Sometimes equivalent to supremal global support

Algorithm for supremal local support

- Based on the communication graphs of the problem
- Message passing algorithm
- Communication graphs coincide with trees : supremal global support





Outline

Research plan

First step

Implement algebraic computations on languages in terms of automata.

Second step

Extend to distributed/modular search for optimal plans.

Third step ?

Could we parallelize the A^* algorithm and compute an optimal and sequential plan in a distributed way ?
Is it also possible for a partially ordered plan ?