

# From one to many: Multiagent Planning

Ronen Brafman, Carmel Domshlak, Raz Nissim

(Antonín Komenda, [komenda@agents.fel.cvut.cz](mailto:komenda@agents.fel.cvut.cz))

## Overview

- systems consisting of agents
- an **agent** is a bounded entity
- the entities **interact** with each other
- generally no limitations on what an agent is (robots, humans, programs, ...)

## Overview

- systems consisting of agents
- an **agent** is a bounded entity
- the entities **interact** with each other
- generally no limitations on what an agent is (robots, humans, programs, ...)

## For scope of this lecture

- agents  $\sim$  intelligent programs
- interaction  $\sim$  message passing

## Technically

- agents  $\sim$  a computational thread or process running ideally on its own processor (core)
- interaction  $\sim$  inter-process sending of messages (potentially over network)

# Constraint Satisfaction Problems (CSPs)

## What is a CSP?

- finite set of variables  $v_1, v_2, \dots, v_k$
- non-empty domain of possible values for each variable  
 $D_{v_1}, D_{v_2}, \dots, D_{v_k}$
- finite set of constraints  $C_1, C_2, \dots, C_m$
- each constraint  $C_i$  specifies allowable combinations of values for subsets of variables
- a solution is an assignment of values to all variables that satisfies all constraints

# Constraint Satisfaction Problems (CSPs)

## Example

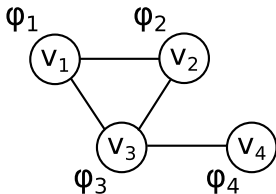
- variables:
  - $v_1 \in \{0, 1, 2\} = D_{v_1}$
  - $v_2 \in \{3, 4\} = D_{v_2}$
  - $v_3 \in \{5, 6\} = D_{v_3}$
  - $v_4 \in \{5, 6\} = D_{v_4}$
- constraints:
  - $C_1 = \{v_1 = 0 \Rightarrow v_2 = 4\}$
  - $C_2 = \{v_1 = 1 \Rightarrow v_2 = 3\}$
  - $C_3 = \{v_2 = 3 \Rightarrow v_3 = 5\}$
  - $C_4 = \{v_3 = 5 \Rightarrow v_1 \neq 0 \wedge v_1 \neq 2\}$
  - $C_5 = \{v_3 = v_4\}$
- solution:
  - $?, +?$

# Distributed Constraint Satisfaction Problems (DisCSPs)

## Distribution of CSP

- each agent  $\varphi_i \in \Phi$  is responsible for one variable  $v_i$
- constraints are over more agents according to their variables
- an **agent interaction graph** of the agents is based on their variables and the constraints

Interaction graph of the previous example:



## Detour to Graph theory

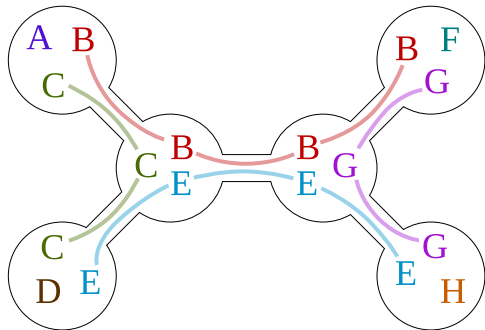
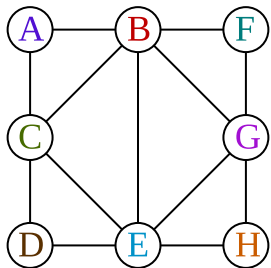
Graph tree-decomposition, width and tree-width



## Definitions

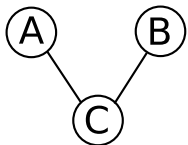
- a **tree-decomposition** of a graph  $G$  is  $(T, W)$ , where  $T$  is a tree and  $W = (W_t : t \in V(T))$  satisfies:
  - $\bigcup_{t \in V(T)} W_t = V(G)$  (each graph vertex is associated with at least one tree node)
  - $\forall uv \in E(G) : \exists t \in V(T)$  s.t.  $u, v \in W_t$  (vertices are adjacent in the graph only when the corresponding subtrees have a node in common)
  - if  $t' \in T[t, t'']$  (path), then  $W_t \cap W_{t''} \subseteq W_{t'}$  (the nodes associated with a vertex form a connected subset of  $T$ .)
- the **width** of a graph is  $\max(|W_t| - 1 : t \in V(T))$
- the **tree-width** of  $G$  is the minimum width of a tree-decomposition of  $G$

# Graph tree-decomposition, width and tree-width

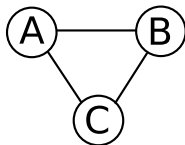


# Graph tree-decomposition, width and tree-width

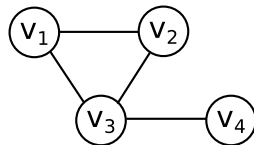
$tw(G)=1$



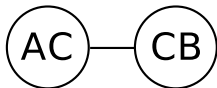
$tw(G)=2$



$tw(G)=2$



$w(G)=1$



$w(G)=2$



homework

# Graph tree-decomposition, width and tree-width

## Formally

- $tw(G) = 1 \Leftrightarrow G$  is forest (or tree or a series) graph
- $tw(G) = 2 \Leftrightarrow G$  is series-parallel graph
- $tw(G_n^{k \times k\text{-grid}}) = k = \sqrt{n}$  for a grid graph of  $n$  vertices  
[bigger homework (optional)]
- $tw(K_n) = n - 1$  for a complete graph of  $n$  vertices

## Informally

- tree-width of a graph determines its “cliquishness” (opposite of linearity or “treeness”)
- in the DisCSP problems the tree-width of the interaction graph is related to coupling of the problem

End of the detour

(not end of the lecture, though)

# Distributed Constraint Satisfaction Problems (DisCSPs)

## Solving (Dis)CSP

- more families of algorithms solving CSP
- *Adaptive Tree Consistency (ATC)* – based on tree-decomposition of the underlying constraint graph, can be described as message passing among the variables (agents)

## Complexity of ATC

- proven that time complexity of ATC is
$$O(kD^{\omega+1})$$
- $k$  corresponds to number of CSP variables and therefore number of agents in DisCSP
- $D = \max_{i=1}^k D_i$ , i.e., size of the largest domain (because of asymptotic complexity)
- $\omega$  is tree-width of the constraint graph (corresponds to the agent interaction graph)

## Planning for loosely coupled multiagent systems

[R. Brafman, C. Domshlak: From one to many: Planning for loosely coupled multiagent systems, In *Proceedings of ICAPS'08*, 2008]

# Motivation

## Logistics planning

Deliver packages using vehicles (trucks, airplanes, ships) operating in/between different countries/regions/cities

- Classical benchmark for “single-agent” planning
- Classic example of a **distributed system**  $\rightsquigarrow$  **vehicle = agent**

## (Informal) Question

Can we exploit the fact that the domain describes a naturally distributed system to make planning more efficient?

## (Ultimate) Answer

YES, we can solve distributed components independently



# Motivation

## Logistics planning

Deliver packages using vehicles (trucks, airplanes, ships) operating in/between different countries/regions/cities

- Classical benchmark for “single-agent” planning
- Classic example of a **distributed system**  $\rightsquigarrow$  **vehicle = agent**

## (Informal) Question

Can we exploit the fact that the domain describes a naturally distributed system to make planning more efficient?

## (Ultimate) Answer

YES, we can solve distributed components independently

# Basic Motivation/Intuition

$k$ -agents MA Systems (Logistics domain example)

## Fully decoupled

Vehicles are a priori responsible for different packages

Same as planning  $k$  times for a single agent

→ linear time-complexity growth  
( $\exp(k)$  time-complexity reduction)

## Fully coupled

Vehicles have to move the same packages and maybe coordinate on loads/unloads

Same as planning for a single “ $k$ -times larger” agent

→  $\exp(k)$  time-complexity growth  
(no reduction in time-complexity)

# Main Ideas

## A New Graphical Model

Potential (positive and negative) interactions between the agents' individual abilities (= actions)

## System coupling-level

Define an **interaction graph** of the system

**Nodes** = agents

**Edges** = agents need (to coordination with) each other

Parameter  $\omega \rightsquigarrow$  **tree-width** of interaction graph

# Main Ideas

## A New Graphical Model

Potential (positive and negative) interactions between the agents' individual abilities (= actions)

## System coupling-level

Parameter  $\omega \rightsquigarrow$  **tree-width** of interaction graph

## Problem coupling-level

Some problems require more coordination than others!

Parameter  $\delta \rightsquigarrow$  **minmax** number of times a single agent needs some other agent to do something for it

# Main Ideas

## System coupling-level

Parameter  $\omega \rightsquigarrow$  **tree-width** of interaction graph

## Problem coupling-level

Parameter  $\delta \rightsquigarrow$  **minmax** number of times a single agent needs some other agent to do something for it

## Algorithm

- Fix the agents' commitments to each other  
 $\rightsquigarrow$  *careful selection of language matters!*
- Let each agent **independently** plan “in-between” commitments
- Use iterative deepening to extend the number of **per-agent** commitments if needed

# Agent Actions

## Logistics planning

Deliver packages using vehicles (trucks, airplanes, ships) operating in/between different countries/regions/cities

- Actions  $\text{move}(v, \text{from}, \text{to}), \text{load}(p, v, \text{at}), \text{unload}(p, v, \text{at})$
- Agents: vehicles
- Vehicle agent actions:  
moving it, loading into it, unloading from it

## From STRIPS to MA-STRIPS

Everything is the same, except that  
**actions are partitioned** between the agents

# From STRIPS to MA-STRIPS

## Definition

A STRIPS problem is given by a quadruple  $\Pi = \langle P, A, I, G \rangle$ , where:

- $P$  is a finite set of *atoms*,  $I \subseteq P$  is the *initial situation*, and  $G \subseteq P$  encodes the *goal* situations,
- Each action  $a \in A$  is given by  $\langle \text{pre}(a), \text{add}(a), \text{del}(a) \rangle$ .

# From STRIPS to MA-STRIPS

## Definition

An MA-STRIPS problem for a system of agents  $\Phi = \{\varphi_i\}_{i=1}^k$  is given by a quadruple  $\Pi = \langle P, \{A_i\}_{i=1}^k, I, G \rangle$ , where:

- $P$  is a finite set of *atoms*,  $I \subseteq P$  is the *initial situation*, and  $G \subseteq P$  encodes the *goal situations*,
- For  $1 \leq i \leq k$ ,  $A_i$  is the set of actions that the agent  $\varphi_i$  is capable of performing. Each action  $a \in A = \bigcup A_i$  is given by  $\langle \text{pre}(a), \text{add}(a), \text{del}(a) \rangle$ .



# Solving MA-STRIPS Problems

## Standard Approaches

- 1 Compile into a single-agent STRIPS problem
  - ☹ Lose all structure and obtain  $k$ -times larger “agent”
  - ☹ Worst-case complexity exponential in description size or shortest plan (depending on search strategy)
- 2 Try to solve as much as possible locally and compose the resulting individual agent plans
  - ☹ What can we say about it?

# Solving MA-STRIPS Problems

## Standard Approaches

- 1 Compile into a single-agent STRIPS problem
  - ☹ Lose all structure and obtain  $k$ -times larger “agent”
  - ☹ Worst-case complexity exponential in description size or shortest plan (depending on search strategy)
- 2 Try to solve as much as possible locally and compose the resulting individual agent plans
  - ☹ What can we say about it?

# A Closer Look at Agent Actions

## Private vs. Non-Private

**Private** affect and depend only on that agent

**Non-Private** all the rest

## Logistic planning

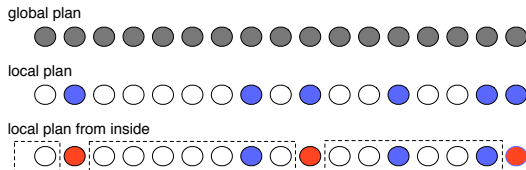
- Move actions are private  
(influence and influenced only by vehicle location)
- Loading into/unloading from a vehicle is non-private  
~> except if the package location is private to the vehicle!

# A Closer Look at Agent Subplans

## Private vs. Non-Private

**Private** affect and depend only on that agent

**Non-Private** all the rest

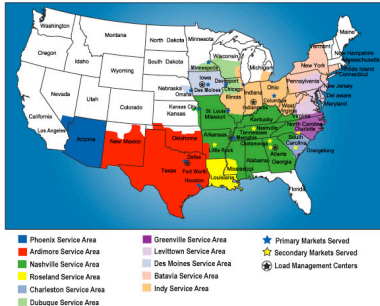


- **non-private actions** in the plan  $\rightsquigarrow$  **coordination points**
- **arbitrarily long** sequences of private actions between adjacent non-private actions

# Example: Logistics

## Logistics

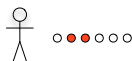
- imagine vehicles moving on a large map
- each vehicle has a **service region**
- ↪ between each load/unload action, there are multiple move actions by the vehicle



# Main Idea

## “Algorithm”

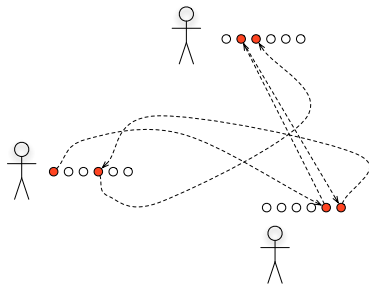
- 1 Find a good choice of coordination points
- 2 Solve  $k$  local planning problems over the private actions of the agents only



# Main Idea

## “Algorithm”

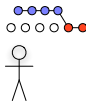
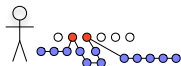
- 1 Find a good choice of coordination points
- 2 Solve  $k$  local planning problems over the private actions of the agents only



# Main Idea

## “Algorithm”

- 1 Find a good choice of coordination points
- 2 Solve  $k$  local planning problems over the private actions of the agents only

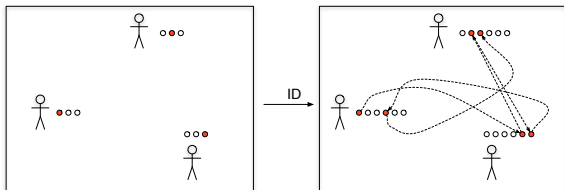




# Main Idea

## “Algorithm”

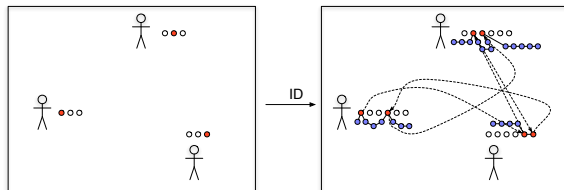
- 1 Find a good choice of coordination points
  - **Iterative deepening** on  $\delta$  — # of coord-points **per agent**
  - For each choice of  $\delta$ 
    - Define a **CSP** whose solutions are consistent assignments to the coordination points
- 2 Solve  $k$  local planning problems over the private actions



# Main Idea

## “Algorithm”

- 1 Find a good choice of coordination points
- 2 Solve  $k$  local planning problems over the private actions
  - purely **independent** phase  $\leadsto$  unary constraints
  - can be reduced to **regular STRIPS** planning



The complexity is derived from

- number of agents  $k$  in set  $\Phi$  with public actions  $A_i^{pub}$
- maximal complexity of the local planning  $\mathcal{I}$  with a cost function for switching from regular planning  $f(\cdot)$
- number of “coordination” CSPs we have to solve (corresponds to  $\delta$ )
- solving each “coordination” CSP  $O(kD^{\omega+1})$
- length of the “coordination” plan  $k\delta$

The idea of  $k\delta$ :

$$\begin{array}{l} \alpha : \\ \beta : \end{array} \left( \begin{array}{cccccc} a_1^\alpha & * & a_2^\alpha & * & a_3^\alpha & * \\ * & a_1^\beta & * & a_2^\beta & * & a_3^\beta \end{array} \right).$$

Size of agent's domain is:

$$|D_i| = \sum_{d=1}^{\delta} \binom{k\delta}{d} \cdot |A_i^{pub}|^d = O((k\delta|A_i^{pub}|)^{\delta+1}).$$

## Terms

- $\binom{k\delta}{d}$  represents all possible **combinations** of  $d$  virtual time points for the public actions (e.g., for  $d = 2, k\delta = 6$  there are 15 of them  $\{(1, 2), (1, 3), \dots, (1, 6), (2, 3), (2, 4), \dots, (5, 6)\}$ )
- $|A_i^{pub}|^d$  represents **all possible** public action sequences of length  $d$  (e.g, for  $d = 2$  and  $|A_i^{pub}| = 2$  they are  $\{a_1 a_1, a_1 a_2, a_2 a_1, a_2 a_2\}$ )
- the summed up result represent the number of all possible coordination sequences for one agent.

Time complexity of the unary **internal-planning constraints**:

$$O(f(\mathcal{I}) \cdot k \cdot \max_{i \in \Phi} |D_i|) = O(f(\mathcal{I}) \cdot k(k\delta |A^{pub}|)^{\delta+1}) = O_{ipc},$$

## Terms

- $f(\mathcal{I}) \cdot k$  the internal planning has to be run by each agent
- asymptotically (in worst case)  $\max_{i \in \Phi} |D_i|$  domains has to be planned by all agents
- asymptotically (in worst case)  $|A_i^{pub}|$  for all agents are all public actions  $|A^{pub}|$

Time complexity of the **coordination constraints**:

$$O(k \cdot \max_{i \in \Phi} |D_i|^{\omega+1}) = O(k(k\delta |A^{pub}|)^{\delta\omega+\epsilon}) = O_{cc},$$

## Terms

- based on ATC algorithm time complexity  $O(kD^{\omega+1})$
- $D = \max_{i \in \Phi} |D_i|$
- $\epsilon = \delta + \omega + 1$  is dominated by  $\delta\omega$
- asymptotically (in worst case)  $|A_i^{pub}|$  for all agents are all public actions  $|A^{pub}|$

# Final Complexity

Time final time complexity bound of the **multiagent planning**:

$$O_{ipc} + O_{cc} = O(f(\mathcal{I}) \cdot k(k\delta|A^{pub}|)^{\delta+1} + k(k\delta|A^{pub}|)^{\delta\omega+\epsilon}).$$

The exponential bounds can be therefore expressed as:

$$f(\mathcal{I}) \cdot \exp(\delta) + \exp(\delta\omega)$$

Exponentially does not depend on

Algorithm complexity has no direct exponential dependence on the **number of agents**  $k$ , has no direct exponential dependence on the **length of the individual plans** of the agents and has no direct exponential dependence on the **size of the original planning problem**.

Exponentially depends on

Algorithm complexity is exponentially dependent on **number of coordination points**, i.e., length of the coordination plan and on **tree-width of the agent interaction graph**.

# Multiagent A\*

[R. Nissim, R. Brafman: Multi-Agent A\* for Parallel and Distributed Systems,  
In *Proceedings of HDIP Workshop (ICAPS)*, 2012]



## Overview

- based on partition of actions from the previous slides
- private/public actions (respecting privacy)
- A\* expansion only of agent's own actions
- distributed optimal search
- distributed termination detection
- ~~currently most efficient distributed planning approach (2012)~~  
~> 2015  
PSM planner (winner of the Competition of DMAP planners)

## Our approach – optimal forward search

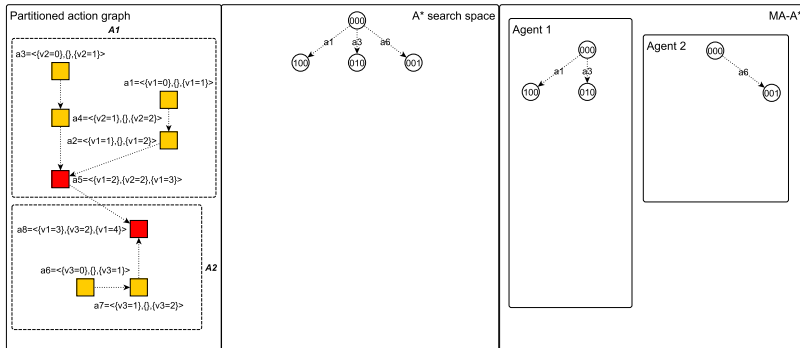
Each agent runs an A\*-like search separately, using its own open/closed list. In each iteration, the agent performs:

### MA-A\*

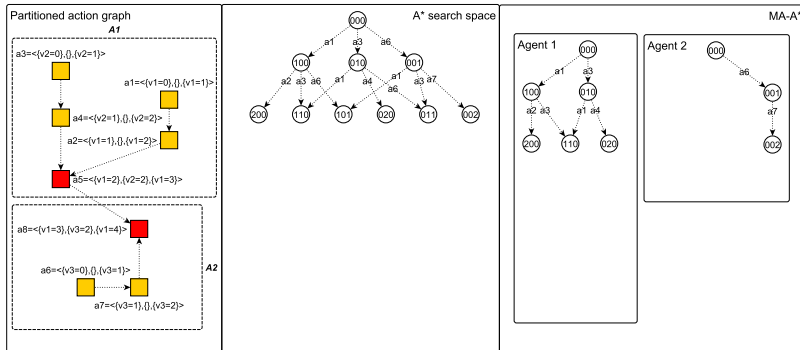
- Receive messages and insert states into open list.
- Retrieve first node  $n$  from open list.
- If  $n$  is a solution, perform distributed optimality check.
- Expand  $n$  using the agent's own actions only.
- Compute  $h$ -value and add to open list all children  $n'$ .
- If  $n'$  was obtained by applying a public action
  - then send  $n'$  to all agents to which  $n'$  is relevant.

Messages contain the full state  $n'$ , its  $g$  and  $h$ -values, and its creating action.

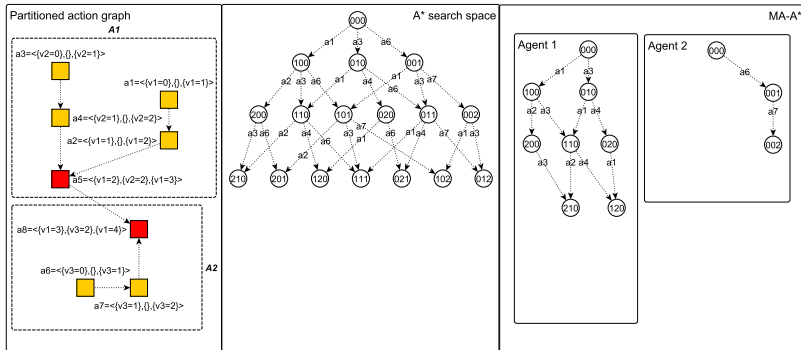
# Running example



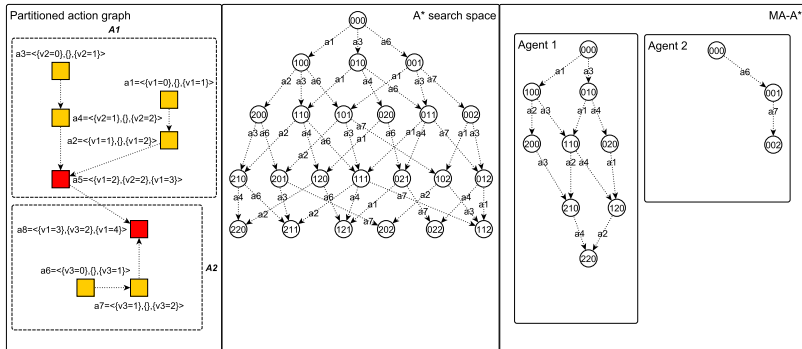
# Running example



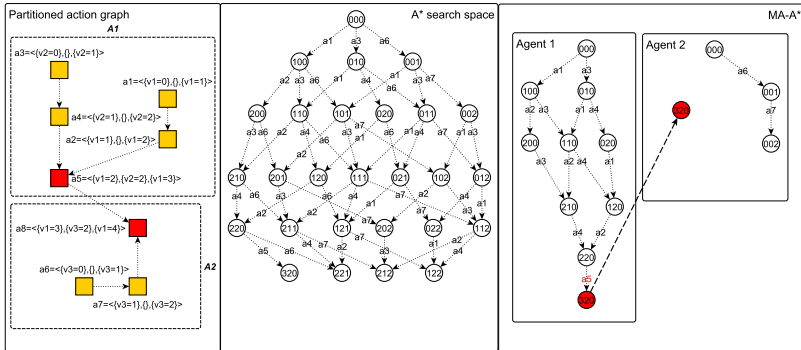
# Running example



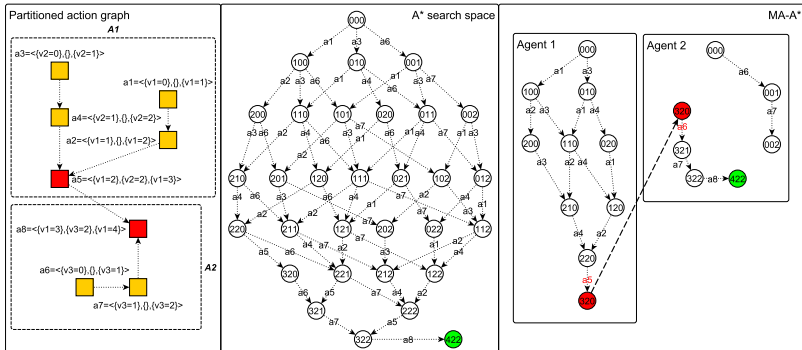
# Running example



# Running example



# Running example





# Relevancy of messages

- A state  $s$  is relevant to an agent if it has a public action for which all public preconditions hold in  $s$ .
- When some agent performs a private action, other agents' view of the system relevant to them has not changed!
- Sending only states for which the creating action is public, maintains optimality.
- This effectively prunes many equivalent parts of the search space  $\implies$  **may result in fewer expansions than centralized A\***.

# Experimental results

Problem (# of processors)	# agents	LM-cut heuristic					Merge&Shrink heuristic					Planning First	
		A*		MAP-A*			A*		MAD-A*			time	$\delta$
		time	expands	time	expands	eff.	time	init-h	time	init-h	eff.		
logistics7-1(4)	4	55.3	155289	27	504102	0.51	0	44	57	36	0	-	N/A
logistics8-1(4)	4	24.1	43665	13	168545	0.46	0.1	44	49.5	37	0	-	N/A
logistics10-0(4)	5	203	193846	66.6	627314	0.76	81.7	43	-	36	0	-	N/A
Rovers3(2)	2	0	50	0	90	1.00	0	9	0	6.5	1.00	0.3	2
Rovers4(2)	2	0	9	0.04	68	0	0	8	0	6	1.00	0.2	1
Rovers5(2)	2	8.8	37397	1.8	18975	2.44	11.7	20	4	10.5	1.46	9.3	3
Rovers6(2)	2	-	-	236	2255393	$\infty$	-	27	325	17.5	$\infty$	-	N/A
Rovers7(3)	3	6.7	18315	1	12929	2.23	55.9	14	7.2	9	2.59	38.5	3
Rovers8(4)	4	-	-	154	1271971	$\infty$	-	15	-	10	N/A	-	N/A
Rovers12(4)	4	12.1	15222	0.9	10704	3.36	119	16	22.2	8.25	1.34	-	N/A
Rovers14(4)	4	-	-	598	5311741	$\infty$	-	17	-	11	N/A	-	N/A
satellites5(3)	3	1.3	1174	0.1	793	4.33	7	13	3.8	8	0.61	52.3	2
satellites6(3)	3	3.5	2976	0.2	1650	5.83	23.5	18	9.2	9.3	0.85	457	3
satellites7(4)	4	94.5	36652	12.4	53465	1.91	-	14	343	9.5	$\infty$	-	N/A
satellites8(4)	4	-	-	94.8	345667	$\infty$	-	15	-	10	N/A	-	N/A
satellites9(4)	5	-	-	105	2132756	$\infty$	-	18	-	11	N/A	-	N/A
satellites10(4)	5	-	-	61.8	95192	$\infty$	-	17	-	10	N/A	-	N/A
zenotravel9(3)	3	72.1	15408	20	29321	1.20	56.7	19	370	14	0.05	-	N/A
zenotravel10(3)	3	16.1	1587	4.3	3453	1.25	26.7	22	-	17	0	-	N/A
zenotravel12(3)	3	458	41311	57	41819	2.68	-	-	-	-	N/A	-	N/A
zenotravel13(3)	3	-	-	382	185827	$\infty$	-	-	-	-	N/A	-	N/A

**Experimental results:** Comparison of centralized A\*, MA-A\* in its parallel (MAP-A\*) and distributed (MAD-A\*) settings, and Planning First. Runtime (in sec.), number of expanded nodes and efficiency values (speedup/# of processors) are shown.

# **FF Heuristic in Multiagent Planning**

Build RPGs using only each agent's actions - **incomplete**

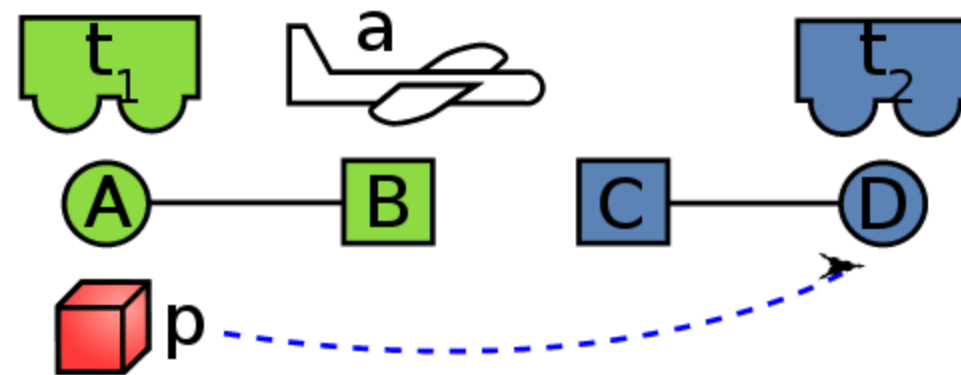
So use also projections of other agent's actions - **complete, but bad estimates**

Let's build the RPGs and extract the heuristic distributedly - **equal to global heuristic**

(proof sketches in the paper)

# The Algorithm

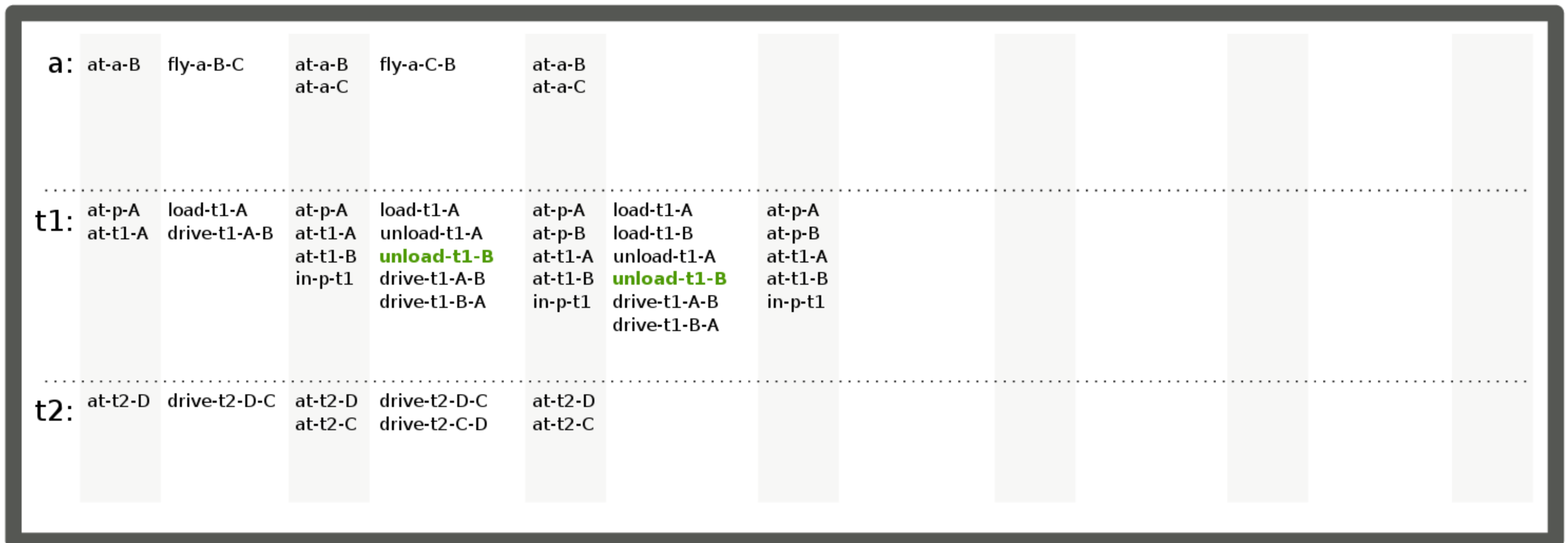
Explained on the following logistics example:



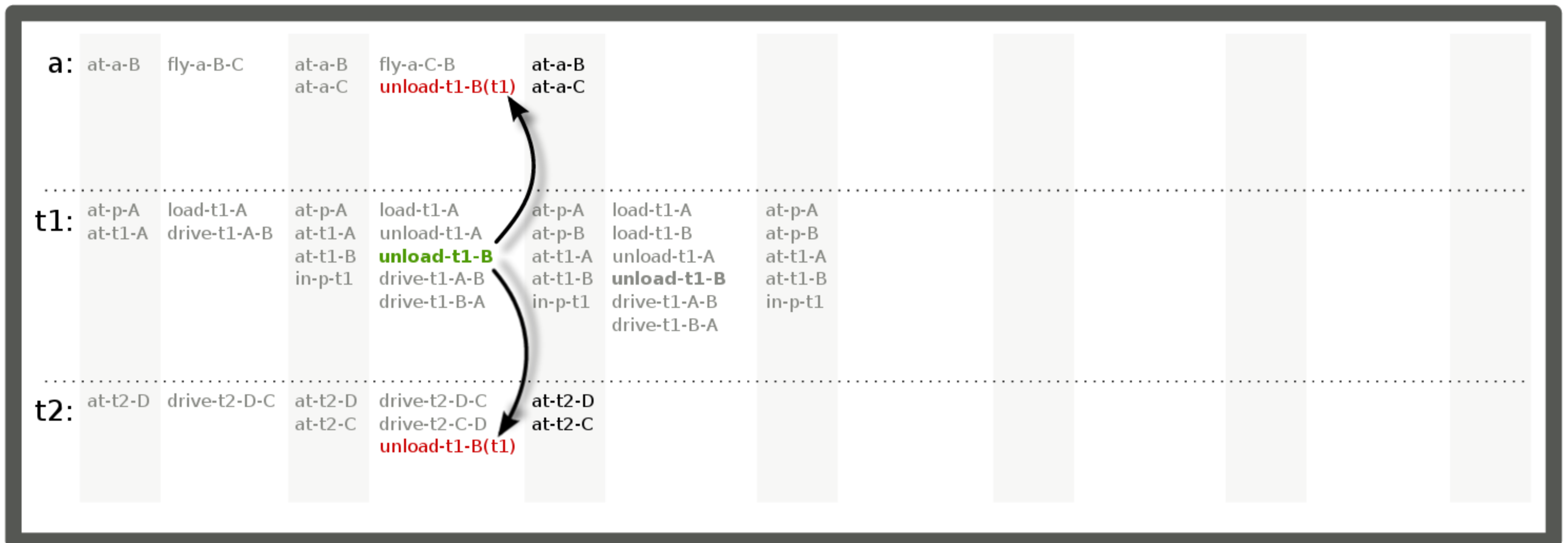
The following table show each agent's RPG:

a:									
t1:									
t2:									

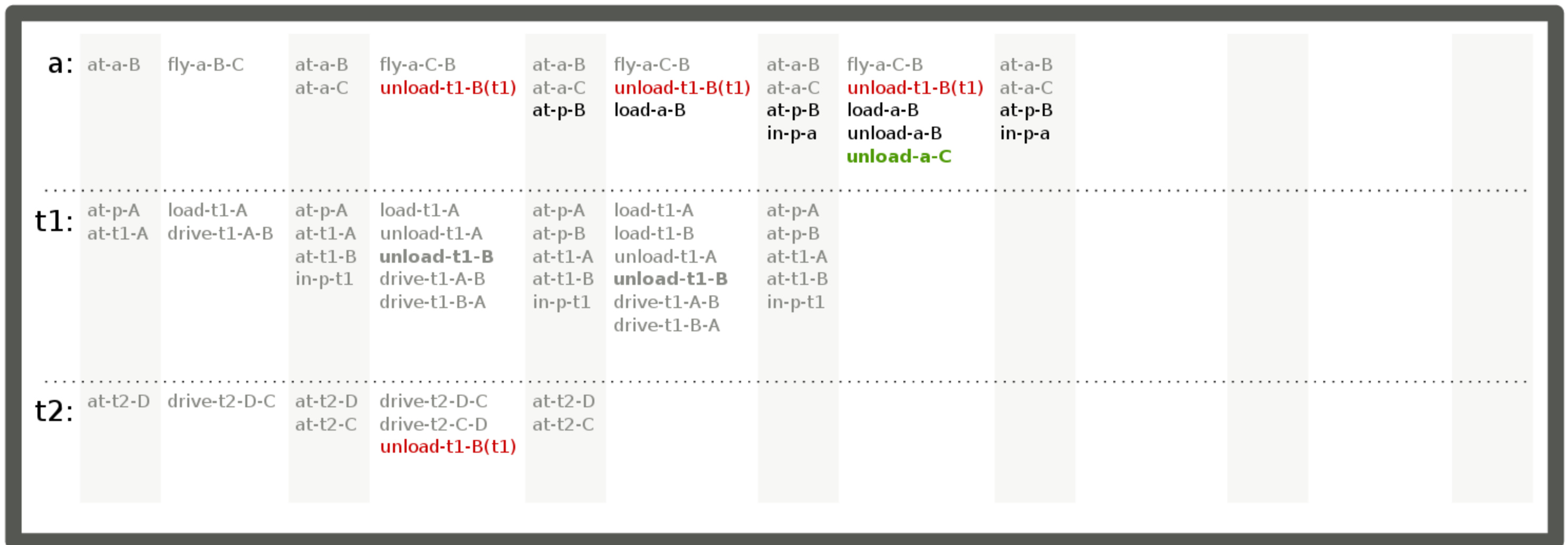
1. **BUILD**: The RPGs are initialized using only respective agent's actions.



## 2.EXTEND: Broadcast public actions and add projections to other agent's RPGs.

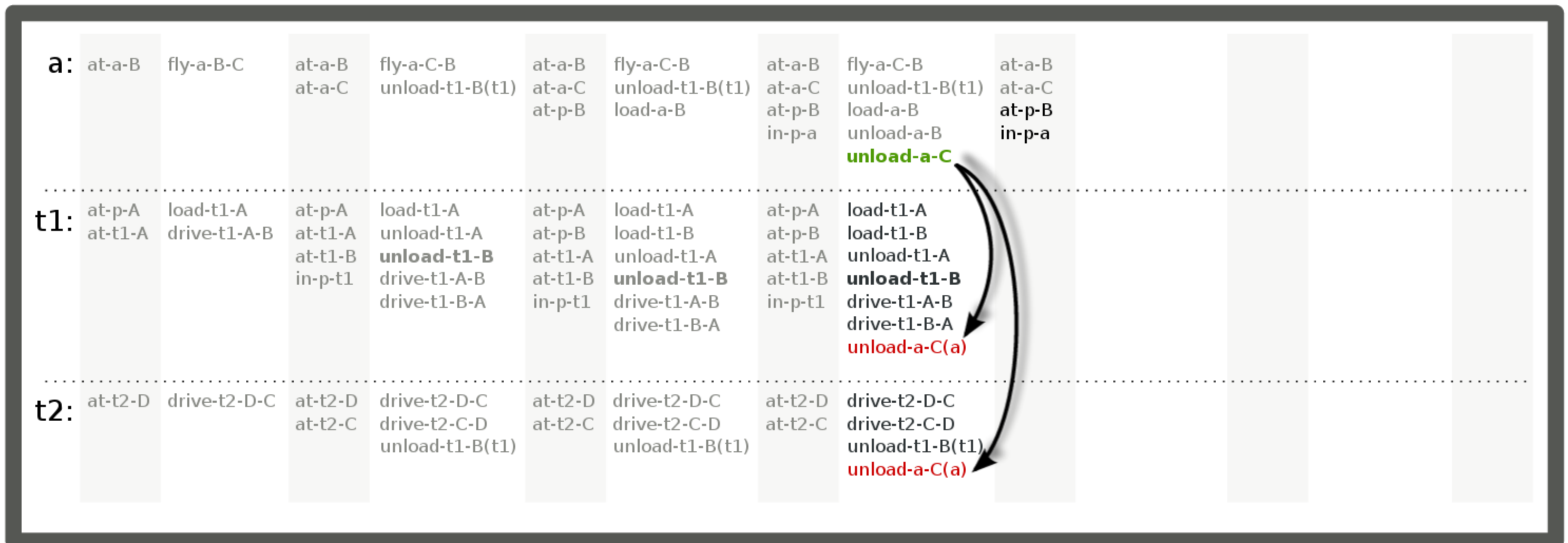


### 3. BUILD: RPGs are updated using newly reachable actions.





#### 4. EXTEND: Newly reachable public actions are again broadcasted.



## 5. BUILD: Another building phase.

<b>a:</b>	at-a-B	fly-a-B-C	at-a-B at-a-C	fly-a-C-B unload-t1-B(t1)	at-a-B at-a-C at-p-B	fly-a-C-B unload-t1-B(t1) load-a-B	at-a-B at-a-C at-p-B in-p-a	fly-a-C-B unload-t1-B(t1) load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a				
<b>t1:</b>	at-p-A at-t1-A	load-t1-A drive-t1-A-B	at-p-A at-t1-A at-t1-B in-p-t1	load-t1-A unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1				
<b>t2:</b>	at-t2-D	drive-t2-D-C	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D unload-t1-B(t1)	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D unload-t1-B(t1)	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D unload-t1-B(t1) <b>unload-a-C(a)</b>	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D at-p-C unload-t1-B(t1) load-t2-C	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D at-p-C in-p-t2 <b>unload-a-C(a)</b> load-t2-C <b>unload-t2-D</b>	at-t2-D at-t2-C at-p-C in-p-t2

## 6. EXTEND: Goal reaching actions are always public and therefore broadcasted.

<b>a:</b>	at-a-B	fly-a-B-C	at-a-B at-a-C	fly-a-C-B unload-t1-B(t1)	at-a-B at-a-C at-p-B	fly-a-C-B unload-t1-B(t1) load-a-B	at-a-B at-a-C at-p-B in-p-a	fly-a-C-B unload-t1-B(t1) load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a	fly-a-C-B unload-t1-B(t1) load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a	fly-a-C-B unload-t1-B(t1) load-a-B unload-a-B <b>unload-a-C</b> <b>unload-t2-D(t2)</b>	at-a-B at-a-C at-p-B in-p-a <b>at-p-D</b>
<b>t1:</b>	at-p-A at-t1-A	load-t1-A drive-t1-A-B	at-p-A at-t1-A at-t1-B in-p-t1	load-t1-A unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A unload-a-C(a)	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A unload-a-C(a)	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A unload-a-C(a) <b>unload-t2-D(t2)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1 <b>at-p-D</b>
<b>t2:</b>	at-t2-D	drive-t2-D-C	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D unload-t1-B(t1)	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D unload-t1-B(t1)	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D unload-t1-B(t1) unload-a-C(a)	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D unload-t1-B(t1) unload-a-C(a)	at-t2-D at-t2-C at-p-C	drive-t2-D-C drive-t2-C-D unload-t1-B(t1) unload-a-C(a) load-t2-C	at-t2-D at-t2-C at-p-C in-p-t2 load-t2-C <b>unload-t2-D</b>



# Distributed FF Heuristic Extraction

Now we will extract the relaxed plan starting from agent t1:

<b>a:</b>	at-a-B	fly-a-B-C	at-a-B at-a-C	fly-a-C-B <b>unload-t1-B(t1)</b>	at-a-B at-a-C at-p-B	fly-a-C-B <b>unload-t1-B(t1)</b> load-a-B	at-a-B at-a-C at-p-B in-p-a	fly-a-C-B <b>unload-t1-B(t1)</b> load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a	fly-a-C-B <b>unload-t1-B(t1)</b> load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a	fly-a-C-B <b>unload-t1-B(t1)</b> load-a-B unload-a-B <b>unload-a-C</b> <b>unload-t2-D(t2)</b>	at-a-B at-a-C at-p-B in-p-a <b>at-p-D</b>
<b>t1:</b>	at-p-A at-t1-A	load-t1-A drive-t1-A-B	at-p-A at-t1-A at-t1-B in-p-t1	load-t1-A unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b> <b>unload-t2-D(t2)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1 <b>at-p-D</b>
<b>t2:</b>	at-t2-D	drive-t2-D-C	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b>	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b>	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b>	at-t2-D at-t2-C at-p-C	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b> load-t2-C	at-t2-D at-t2-C at-p-C in-p-t2	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b> load-t2-C <b>unload-t2-D</b>	at-t2-D at-t2-C at-p-C in-p-t2

# 1.Extracted (backward) first action layer.

<b>a:</b>	at-a-B fly-a-B-C	at-a-B at-a-C fly-a-C-B <b>unload-t1-B(t1)</b>	at-a-B at-a-C at-p-B fly-a-C-B load-a-B <b>unload-t1-B(t1)</b>	at-a-B at-a-C at-p-B in-p-a fly-a-C-B load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a fly-a-C-B load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a fly-a-C-B load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a fly-a-C-B load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a fly-a-C-B load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a fly-a-C-B load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a fly-a-C-B load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a fly-a-C-B load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a fly-a-C-B load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a fly-a-C-B load-a-B unload-a-B <b>unload-a-C</b>
<b>t1:</b>	at-p-A at-t1-A load-t1-A drive-t1-A-B	at-p-A at-t1-A at-t1-B in-p-t1 load-t1-A unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A	at-p-A at-p-B at-t1-A at-t1-B in-p-t1 load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A	at-p-A at-p-B at-t1-A at-t1-B in-p-t1 load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1 load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1 load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1 load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1 load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1 load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1 load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1 load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1 load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1 load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b>
<b>t2:</b>	at-t2-D drive-t2-D-C	at-t2-D at-t2-C drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b>	at-t2-D at-t2-C drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b>	at-t2-D at-t2-C drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b>	at-t2-D at-t2-C drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b>	at-t2-D at-t2-C drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b>	at-t2-D at-t2-C drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b>	at-t2-D at-t2-C drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b>	at-t2-D at-t2-C drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b>	at-t2-D at-t2-C drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b>	at-t2-D at-t2-C drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b>	at-t2-D at-t2-C drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b>	at-t2-D at-t2-C drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b>

## 2. Request the public action from agent t2.

<b>a:</b>	at-a-B	fly-a-B-C	at-a-B at-a-C	fly-a-C-B <b>unload-t1-B(t1)</b>	at-a-B at-a-C at-p-B	fly-a-C-B <b>unload-t1-B(t1)</b> load-a-B	at-a-B at-a-C at-p-B in-p-a	fly-a-C-B <b>unload-t1-B(t1)</b> load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a	fly-a-C-B <b>unload-t1-B(t1)</b> load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a	fly-a-C-B <b>unload-t1-B(t1)</b> load-a-B unload-a-B <b>unload-a-C</b> <b>unload-t2-D(t2)</b>	at-a-B at-a-C at-p-B in-p-a <b>at-p-D</b>
<b>t1:</b>	at-p-A at-t1-A	load-t1-A drive-t1-A-B	at-p-A at-t1-A at-t1-B in-p-t1	load-t1-A unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b> <b>unload-t2-D(t2)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1 <b>at-p-D</b>
<b>t2:</b>	at-t2-D	drive-t2-D-C	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b>	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b>	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b>	at-t2-D at-t2-C at-p-C	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b> load-t2-C	at-t2-D at-t2-C at-p-C in-p-t2	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b> load-t2-C <b>unload-t2-D</b>	at-t2-D at-t2-C at-p-C in-p-t2

### 3. Agent t2 is extracting next layer, t1 is waiting.

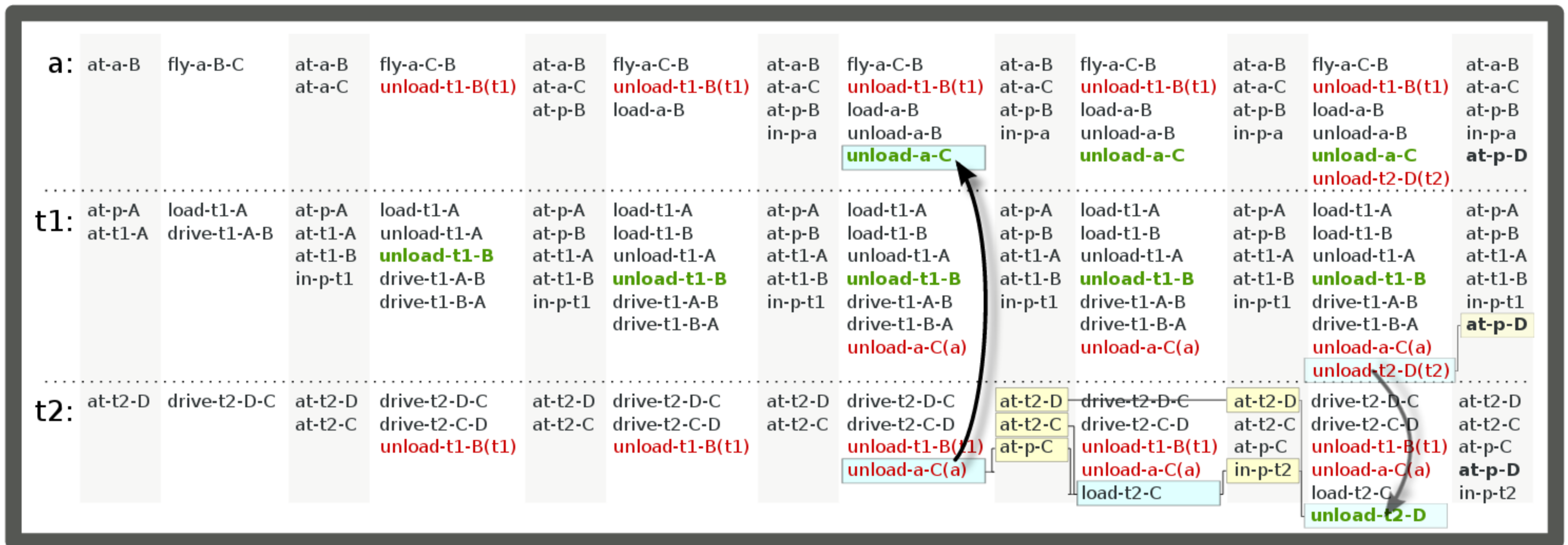
<b>a:</b>	at-a-B	fly-a-B-C	at-a-B at-a-C	fly-a-C-B <b>unload-t1-B(t1)</b>	at-a-B at-a-C at-p-B	fly-a-C-B <b>unload-t1-B(t1)</b> load-a-B	at-a-B at-a-C at-p-B in-p-a	fly-a-C-B <b>unload-t1-B(t1)</b> load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a	fly-a-C-B <b>unload-t1-B(t1)</b> load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a	fly-a-C-B <b>unload-t1-B(t1)</b> load-a-B unload-a-B <b>unload-a-C</b> <b>unload-t2-D(t2)</b>	at-a-B at-a-C at-p-B in-p-a <b>at-p-D</b>
<b>t1:</b>	at-p-A at-t1-A	load-t1-A drive-t1-A-B	at-p-A at-t1-A at-t1-B in-p-t1	load-t1-A unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b> <b>unload-t2-D(t2)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1 <b>at-p-D</b>
<b>t2:</b>	at-t2-D	drive-t2-D-C	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b>	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b>	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b>	at-t2-D at-t2-C at-p-C	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b> load-t2-C	at-t2-D at-t2-C at-p-C in-p-t2	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b> load-t2-C <b>unload-t2-D</b>	at-t2-D at-t2-C at-p-C <b>at-p-D</b> in-p-t2



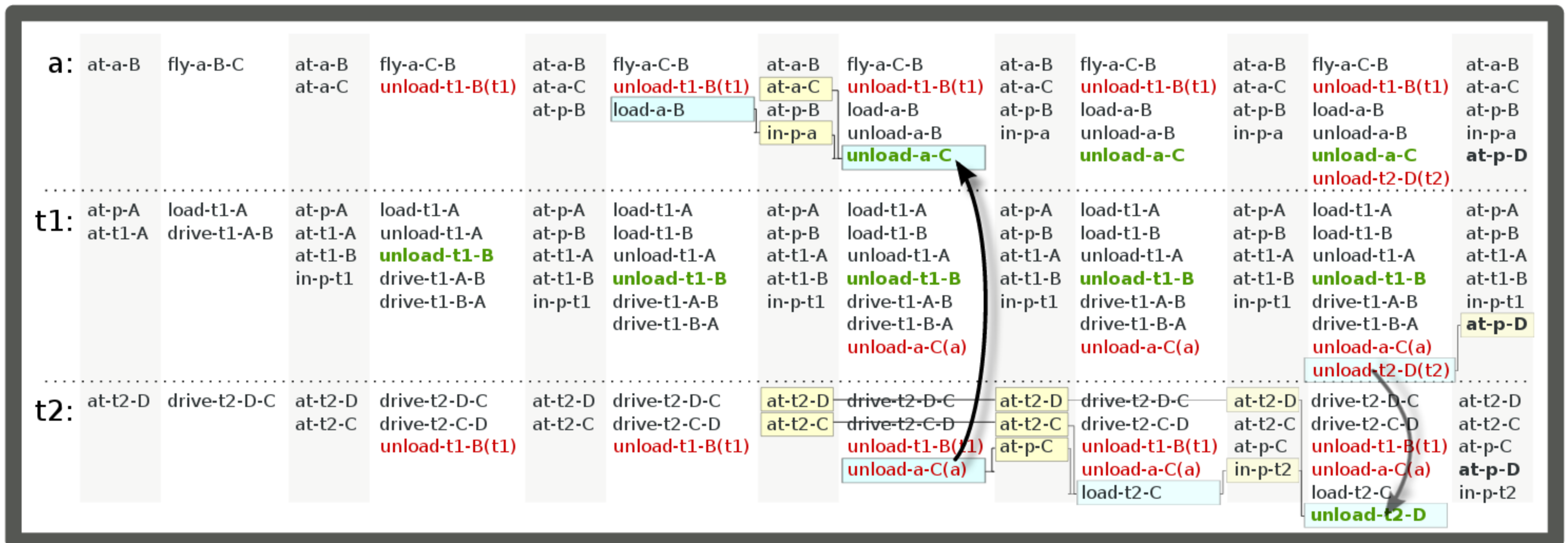
## 4. ...

<b>a:</b>	at-a-B	fly-a-B-C	at-a-B at-a-C	fly-a-C-B <b>unload-t1-B(t1)</b>	at-a-B at-a-C at-p-B	fly-a-C-B load-a-B <b>unload-t1-B(t1)</b>	at-a-B at-a-C at-p-B in-p-a	fly-a-C-B load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a	fly-a-C-B load-a-B unload-a-B <b>unload-a-C</b>	at-a-B at-a-C at-p-B in-p-a	fly-a-C-B load-a-B unload-a-B <b>unload-a-C</b> <b>unload-t2-D(t2)</b>	at-a-B at-a-C at-p-B in-p-a <b>at-p-D</b>
<b>t1:</b>	at-p-A at-t1-A	load-t1-A drive-t1-A-B	at-p-A at-t1-A at-t1-B in-p-t1	load-t1-A unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1	load-t1-A load-t1-B unload-t1-A <b>unload-t1-B</b> drive-t1-A-B drive-t1-B-A <b>unload-a-C(a)</b> <b>unload-t2-D(t2)</b>	at-p-A at-p-B at-t1-A at-t1-B in-p-t1 <b>at-p-D</b>
<b>t2:</b>	at-t2-D	drive-t2-D-C	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b>	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b>	at-t2-D at-t2-C	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b>	at-t2-D at-t2-C at-p-C	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b> load-t2-C	at-t2-D at-t2-C at-p-C in-p-t2	drive-t2-D-C drive-t2-C-D <b>unload-t1-B(t1)</b> <b>unload-a-C(a)</b> load-t2-C <b>unload-t2-D</b>	at-t2-D at-t2-C at-p-C in-p-t2 <b>at-p-D</b>

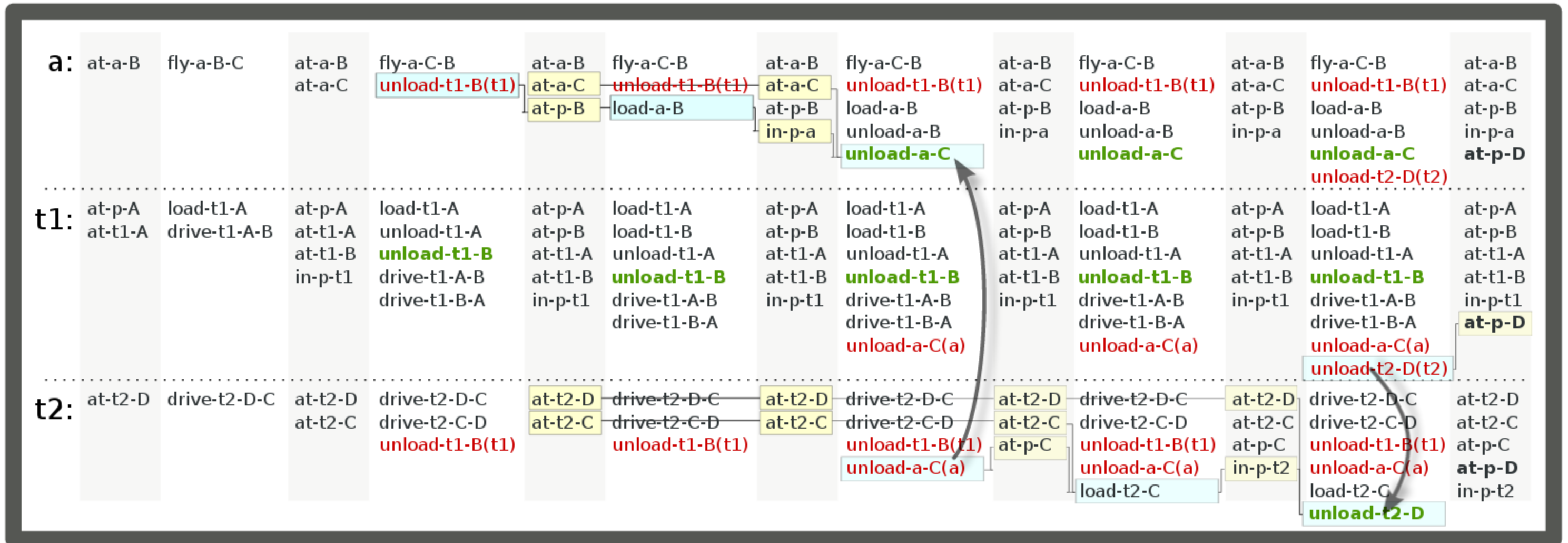
## 5. Agent t2 has reached public action of agent a and sends request.



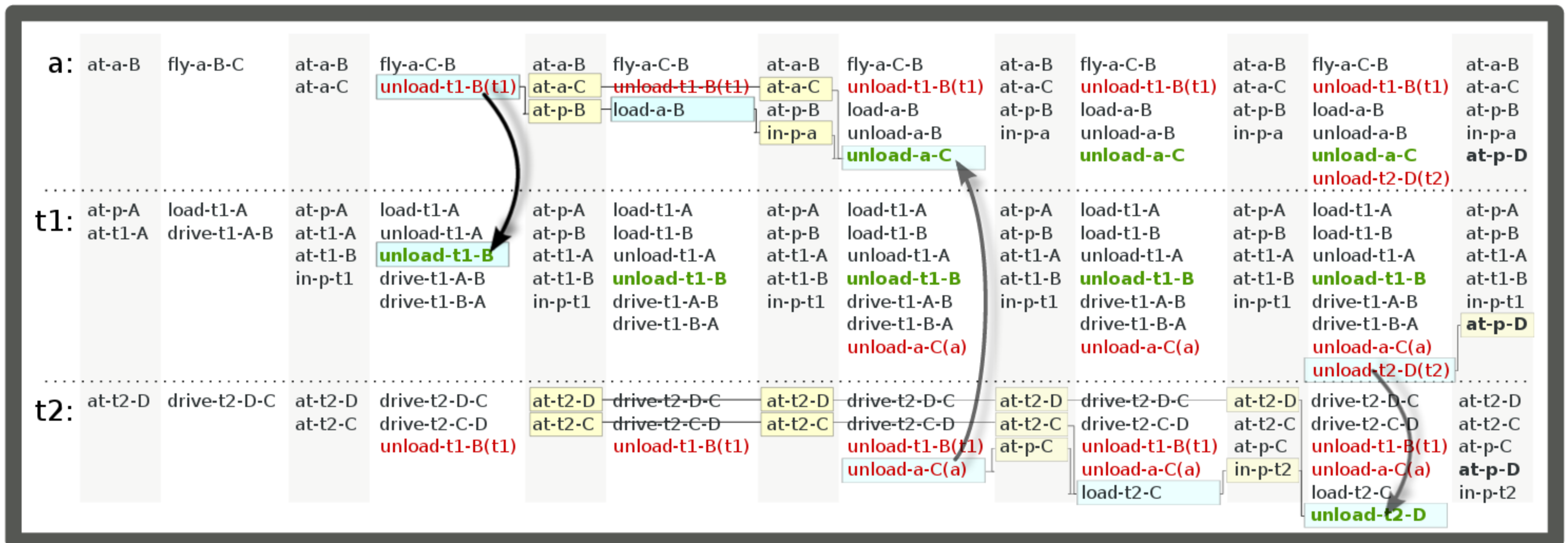
## 6. Both agents t2 and a are building their respective RPs.



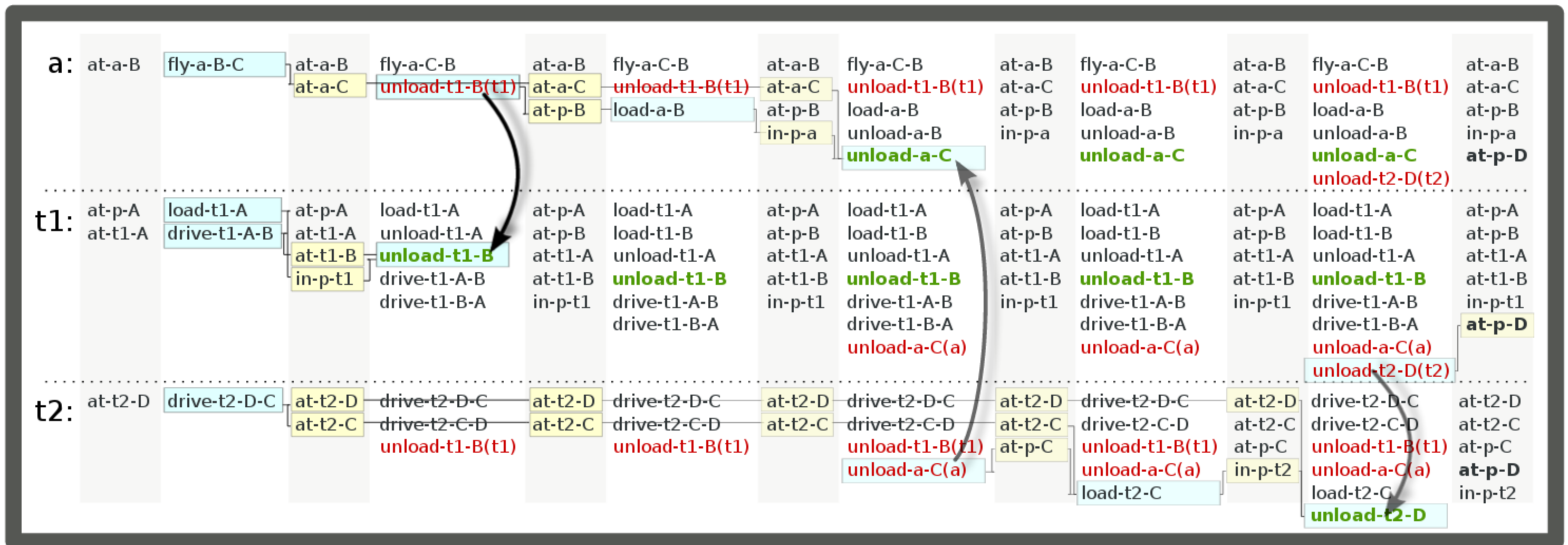
# 7. ...



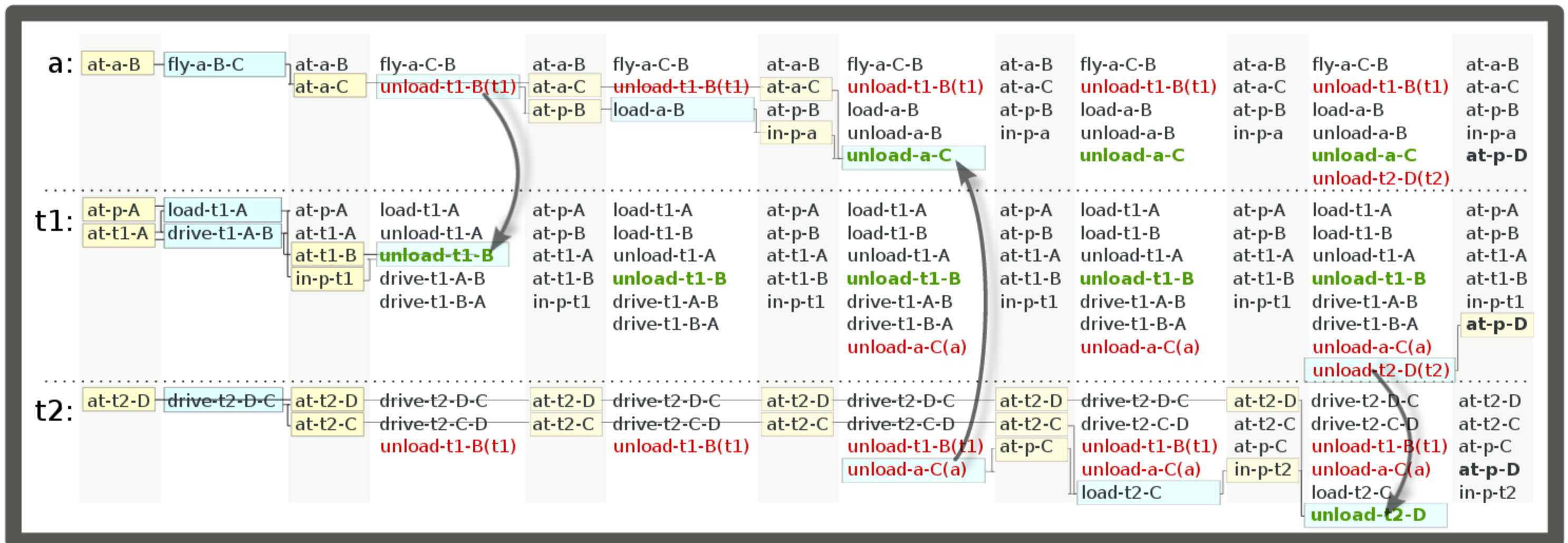
## 8. Agent a requests agent t1, agent t1 starts new RP extraction.



## 9. All agents are extracting the RPs.



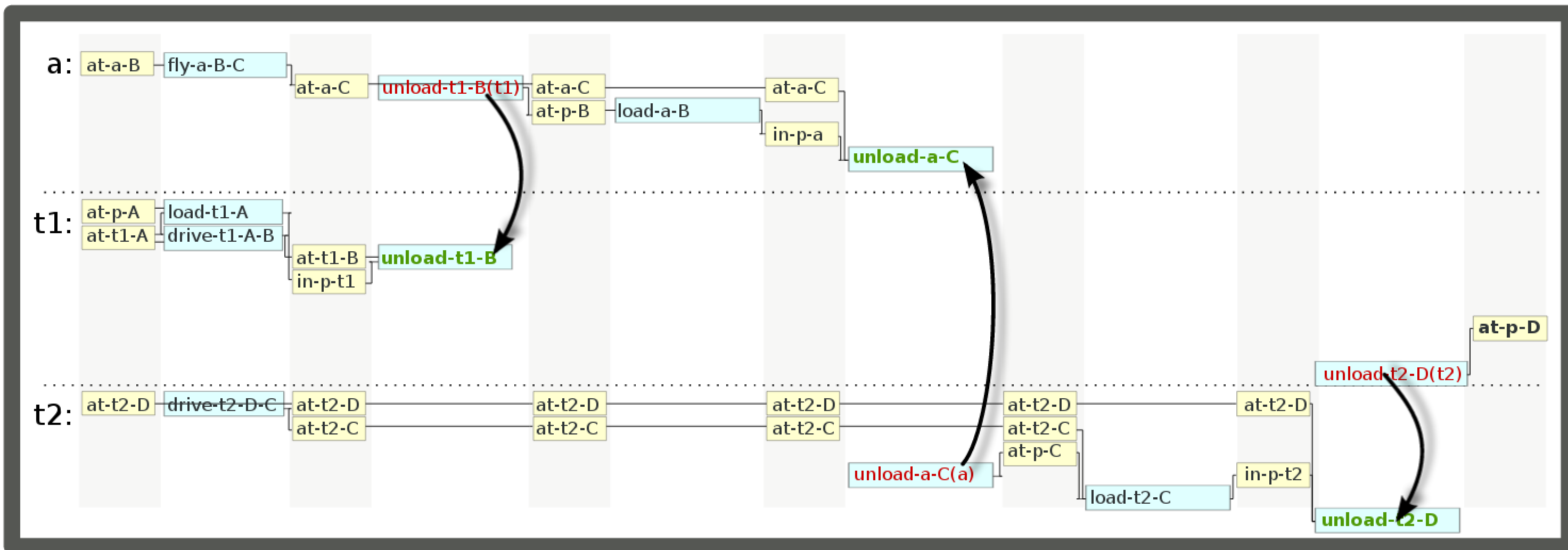
## 10. All agents have reached the initial state.







...



...

