

KRR Project - Documentation

Dmytro Narepekha

Jianhao Luo

Suzzane Abiri

Sabrila Trifebi Shina

Warsaw University of Technology

Faculty of Computer Science and Information Systems

April 18, 2024

Abstract

The aim of the task was to define an action description language \mathcal{ADL}_{AGP} and corresponding query language \mathcal{QL}_{AGP} that will allow describing a specific group of dynamic systems that takes into account agent-action-program related aspects. Accordingly, this contribution consists of two parts - the theoretical description of the task and the documentation of its practical implementation. In the first part, the introduction provides a brief overview of the task, including the assumptions applied to the corresponding group of dynamic systems. Then, the action description language is defined by means of its syntax and semantics. Following that, the query language is discussed in terms of proposed queries and their satisfiability conditions. Lastly, the theory is concluded with descriptions of various examples that provide an understanding of how the proposed languages can be applied in practice. The second part focuses on the description of the technical aspects of the implemented solution – i.e. used technology, implemented interface and the performed tests.

Contents

1	Theoretical description	2
1.1	Introduction	2
1.1.1	Task description	2
1.2	Action description language	2
1.2.1	Syntax	3
1.2.2	Semantics	3
1.3	Query Language	4
1.3.1	Query statements	4
1.3.2	Satisfiability of queries	5
2	Examples of dynamic systems	5
2.0.1	Shipping orders	5
2.0.2	Preparing for holidays	8

1 Theoretical description

1.1 Introduction

Let DS_{AGP} be a class of dynamic systems that are the primary focus of this task. DS_{AGP} fulfills the following assumptions:

- A1. **Inertia law** – the observed changes, after performing a given action, are only the ones that are directly or indirectly induced by the aforementioned action
- A2. **Complete information about all actions and all fluents** – there are no unknown effects of the actions
- A3. **Determinism** – there is only one possible outcome of an action
- A4. **Only sequential actions are allowed** – the parallel actions are out of consideration
- A5. **Characteristics of actions:**
 - ▷ precondition (a set of literals) reflecting a condition under which the action starts and leads to some effect. If a precondition does not hold, then the action is executed with an empty effect
 - ▷ postcondition (a set of literals) reflecting the effect of the action
 - ▷ agent who performs an action
- A6. **State-Dependent Action Constraints for Agents** – In some states (specified by a propositional formula) some actions cannot be performed by specific agents.
- A7. **Partial descriptions** – Partial descriptions of any state of the system (including the initial one) are allowed.

1.1.1 Task description

The task of this project is stated as follows:

Define an action description language ADL_{AGP} for representing dynamic systems of the class specified above, and define the corresponding query language QL_{AGP} which would allow getting answers for the following queries:

- Q1. Does a condition γ (set of literals) hold after performing a program \mathcal{P} ?*
- Q2. Is an agent ag active in executing a program \mathcal{P} ?*

1.2 Action description language

Let ADL_{AGP} be an action description language defined to represent a class of dynamic systems that satisfy the assumptions outlined in the previous section. The following terminology will be used to specify the syntax of the ADL_{AGP} language:

$\Upsilon = (\mathcal{F}, \mathcal{A}_c, \mathcal{A}_g)$ - signature of the language, where:

- \mathcal{F} is a set of fluents
- \mathcal{A}_c is a set of actions
- \mathcal{A}_g is a set of agents

\bar{f} - literal corresponding to the fluent $f \in \mathcal{F}$ or its negation $\neg f$

$\mathcal{P} = ((A_1, X_1), \dots, (A_n, X_n))$ - program, sequence of actions corresponding their agents.

1.2.1 Syntax

Two types of statements were defined for the $\mathcal{ADL}_{\mathcal{AGP}}$ language. The first ones are the **value statements**. They describe the state (more precisely, fluents) that initially holds in the system or holds in the system after performing a particular sequence of actions. The second types of statement are the **effect statements**. They describe how the system's state changes after performing a given action, namely, what state will result from performing a given action under specified preconditions.

1. Value statements

It is important to stress that the second value statement is, in fact, an abbreviation of the first one (i.e. it is not a separate value statement).

1. $\bar{f}_1, \dots, \bar{f}_k$ **after** $(A_1, X_1), \dots, (A_n, X_n)$ (where $f_1, \dots, f_k \in \mathcal{F}$, $A_1, \dots, A_n \in \mathcal{A}_c$, $X_1, \dots, X_n \in \mathcal{A}_g$)

Meaning: $\bar{f}_1, \dots, \bar{f}_k$ holds after performing a set of actions $(A_1, X_1), \dots, (A_n, X_n)$ in the initial state

2. **initially** $\bar{f}_1, \dots, \bar{f}_k$

Meaning: $\bar{f}_1, \dots, \bar{f}_k$ holds in the initial state

2. Effect statements

Similarly to the previous section, the second effect statement is an abbreviation of the first one. The second emphasizes that the action is executed with no specific preconditions that have to be met before its execution.

1. A **causes** $\bar{f}_1, \dots, \bar{f}_k$ by agent **X** if $\bar{g}_1, \dots, \bar{g}_k$ where $f_1, \dots, f_k \in \mathcal{F}$, $g_1, \dots, g_k \in \mathcal{F}$, $A \in \mathcal{A}_c$, $a \in \mathcal{A}_g$

Meaning: If A is performed in any state satisfying $\bar{f}_1, \dots, \bar{f}_k$ by agent a, then in the resulting state $\bar{f}_1, \dots, \bar{f}_k$ holds.

2. A by agent **X** **causes** $\bar{f}_1, \dots, \bar{f}_k$ where $f_1, \dots, f_k \in \mathcal{F}$

Meaning: The performance of action A by agent a leads to the state for which α holds

1.2.2 Semantics

1. Transition function

A **transition function** is defined as a mapping $\Psi: \mathcal{A}_c \times \mathcal{A}_g \times \Sigma \rightarrow \Sigma$.

$\Psi(A, X, \sigma)$ defined for any $A \in \mathcal{A}_c$, for any $\sigma \in \Sigma$ and for any $X \in \mathcal{A}_g$ gives a resulting state after performing the action A from the state σ by agent X.

The aforementioned function can also be generalized to the mapping $\Psi^*: (\mathcal{A}_c \times \mathcal{A}_g)^* \times \Sigma \rightarrow \Sigma$ in a following way:

1. $\Psi^*(\epsilon, \sigma) = \sigma$
2. $\Psi^*(((A_1, X_1), \dots, (A_n, X_n)), \sigma) = \Psi((A_n, X_n), \Psi^*(((A_1, X_1), \dots, (A_{n-1}, X_{n-1})), \sigma))$

However, for the sake of simplification, the Ψ^* will be further denoted as Ψ . The **structure** of a language $\mathcal{ADL}_{\mathcal{CST}}$ is a pair $S = (\Psi, \sigma_0)$ where Ψ is a transition function and

$\sigma_0 \in \Sigma$ is the initial state.

A **state** is defined as a mapping $\sigma: \mathcal{F} \rightarrow \{0, 1\}$, for any $f \in \mathcal{F}$.

If $\sigma(f) = 1$, then it means that f holds in the state σ and it is denoted by $\sigma \models f$.

If $\sigma(f) = 0$, then it means that f doesn't hold in the state σ and it is denoted $\sigma \models \neg f$.

Furthermore, let Σ denote a set of all states.

2. Satisfiability of language statements

Let $S = (\Psi, \sigma_0)$ be a structure for a language $\mathcal{ADL}_{\mathcal{AGP}}$. A statement s is true in S (denoted by $S \models s$), if and only if:

1. s is of the form: $\bar{f}_1, \dots, \bar{f}_k$ **after** $(A_1, X_1), \dots, (A_n, X_n)$

$S \models s$ if and only if for an initial state $\sigma_0 \in \Sigma$ and the set of actions $A_1, \dots, A_n \in \mathcal{A}_c$, the set of agents $X_1, \dots, X_n \in \mathcal{A}_g$ the following holds: $\Psi^*((A_1, X_1), \dots, (A_n, X_n), \sigma_0) \models \bar{f}_1, \dots, \bar{f}_k$

2. s is of the form: A **causes** $\bar{f}_1, \dots, \bar{f}_k$ by agent X if $\bar{g}_1, \dots, \bar{g}_k$

$S \models s$ if and only if for every state $\sigma \in \Sigma$ such that $\sigma \models \bar{g}_1, \dots, \bar{g}_k$ and the action $A \in \mathcal{A}_c$, the agent $X \in \mathcal{A}_g$, the following holds: $\Psi^*(A, X, \sigma) \models \bar{f}_1, \dots, \bar{f}_k$

3. Model of a language

Let D be an action domain (non-empty set of value or effect statements) in the language $\mathcal{ADL}_{\mathcal{AGP}}$ over a signature $\Upsilon = (\mathcal{F}, \mathcal{A}_c, \mathcal{A}_g)$. A structure for this language $S = (\Psi, \sigma_0)$ is a model of D if and only if:

1. for every $s \in D$, $S \models s$
2. for every $A \in \mathcal{A}_c$, for every $\bar{f}_1, \dots, \bar{f}_k, g_1, \dots, g_k \in \mathcal{F}$, for every $\sigma \in \Sigma$, for every $X \in \mathcal{A}_g$ if one of the following conditions holds:

- (a) there is a statement in D of the form A **causes** $\bar{f}_1, \dots, \bar{f}_k$ by agent X if $\bar{g}_1, \dots, \bar{g}_k$ such that $\sigma \not\models \bar{f}_i$ for some $i = 1, \dots, k$
- (b) D does not contain an effect statement A **causes** \bar{f} if $\bar{g}_1, \dots, \bar{g}_k$

then $\sigma \models \bar{f}_1, \dots, \bar{f}_k$ if and only if $\Psi(A, a, \sigma) \models \bar{f}_1, \dots, \bar{f}_k$

1.3 Query Language

Let $\mathcal{QL}_{\mathcal{AGP}}$ be the query language corresponding to the $\mathcal{ADL}_{\mathcal{AGP}}$.

1.3.1 Query statements

There were two types of queries defined for $\mathcal{QL}_{\mathcal{CST}}$, namely the **value queries** and **active queries**. The value queries answer the question of whether a given condition is satisfied after performing a sequence of actions. The active queries evaluate if a given agent is active for execution of sequence of actions.

1. Value queries

The value query is defined in the following way:

$\bar{f}_1, \dots, \bar{f}_k$ **after** $(A_1, X_1), \dots, (A_n, X_n)$

Meaning: does the condition $\bar{f}_1, \dots, \bar{f}_k$ hold after executing the sequence of actions

$(A_1, X_1), \dots, (A_n, X_n)$

2. Active queries

active X in $(A_1, X_1), \dots, (A_n, X_n)$

Meaning: is the agent X active in the program $(A_1, X_1), \dots, (A_n, X_n)$?

1.3.2 Satisfiability of queries

Let D be an action domain. A query Q is a consequence of D , denoted by $D \models Q$, if and only if Q is true in every model $S=(\Psi, \sigma_0)$ of D :

1. Q is of the form: $\bar{f}_1, \dots, \bar{f}_k$ **after** $(A_1, X_1), \dots, (A_n, X_n)$

$D \models Q$ if and only if for any model $S = (\Psi, \sigma_0)$ of D , the following holds
 $\Psi((A_1, X_1), \dots, (A_n, X_n)), \sigma_0 \models \bar{f}_1, \dots, \bar{f}_k$

2. Q is of the form: **active** X in $(A_1, X_1), \dots, (A_n, X_n)$

$D \models Q$ if and only if for any model $S = (\Psi, \sigma_0)$ of D there exists i between 1 and n , such that $\sigma_i \neq \Psi(A_i, X_i, \sigma_{i-1})$ and where $\Psi(A_{i+1}, X_{i+1}, \sigma_i) = \sigma_{i+1}$

2 Examples of dynamic systems

2.0.1 Shipping orders

There is an order that needs to be processed and delivered to the customer. The status of an order can be described by the following literals: received, packed, shipped, delivered and empty. Initially, the order is neither received nor packed, nor shipped, not delivered, and the shipping box for an order is empty. Placing an order makes it received. Packing an order makes it packed and also the shipping box becomes non-empty. Shipping an order makes it shipped and finally delivering it makes it delivered. Each of those aforementioned actions has an associated agent attached in case the initial conditions for performing a specific action are met.

The set of all considered states in this scenario is defined as $\Sigma = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4\}$, where:

$$\begin{aligned}\sigma_0 &= \{\neg\text{received}, \neg\text{packed}, \neg\text{shipped}, \neg\text{delivered}, \text{empty}\} \\ \sigma_1 &= \{\text{received}, \neg\text{packed}, \neg\text{shipped}, \neg\text{delivered}, \text{empty}\} \\ \sigma_2 &= \{\text{received}, \text{packed}, \neg\text{shipped}, \neg\text{delivered}, \neg\text{empty}\} \\ \sigma_3 &= \{\text{received}, \text{packed}, \text{shipped}, \neg\text{delivered}, \neg\text{empty}\} \\ \sigma_4 &= \{\text{received}, \text{packed}, \text{shipped}, \text{delivered}, \neg\text{empty}\}\end{aligned}$$

The Agents to perform in the program are **Operator**, **Courier**. Operator can only perform actions: *Place*, *Pack*. Courier can only perform actions: *Pack*, *Ship*, *Deliver*. Agents perform undefined actions will have no effect. The structure of the scenario is as follows:

initially $\neg\text{received}$
initially $\neg\text{packed}$
initially $\neg\text{shipped}$
initially $\neg\text{delivered}$
initially empty

Place by Operator causes received if \neg received
Pack by Operator causes packed if received, \neg packed
Pack by Courier causes packed if received, \neg packed
Pack by Operator causes \neg empty if received, \neg packed
Pack by Courier causes \neg empty if received, \neg packed
Ship by Courier causes shipped if packed, \neg shipped
Deliver by Courier causes delivered if shipped, \neg delivered

Hence, the following states are considered in this case:

$\Psi((\text{Place}, \text{Operator}), \sigma_0) = \sigma_1$
 $\Psi((\text{Place}, \text{Courier}), \sigma_0) = \sigma_0$
 $\Psi((\text{Pack}, \text{Operator}), \sigma_0) = \sigma_0$
 $\Psi((\text{Pack}, \text{Courier}), \sigma_0) = \sigma_0$
 $\Psi((\text{Ship}, \text{Operator}), \sigma_0) = \sigma_0$
 $\Psi((\text{Ship}, \text{Courier}), \sigma_0) = \sigma_0$
 $\Psi((\text{Deliver}, \text{Operator}), \sigma_0) = \sigma_0$
 $\Psi((\text{Deliver}, \text{Courier}), \sigma_0) = \sigma_0$

$\Psi((\text{Place}, \text{Operator}), \sigma_1) = \sigma_1$
 $\Psi((\text{Place}, \text{Courier}), \sigma_1) = \sigma_1$
 $\Psi((\text{Pack}, \text{Courier}), \sigma_1) = \sigma_2$
 $\Psi((\text{Pack}, \text{Operator}), \sigma_1) = \sigma_2$
 $\Psi((\text{Ship}, \text{Operator}), \sigma_1) = \sigma_1$
 $\Psi((\text{Ship}, \text{Courier}), \sigma_1) = \sigma_1$
 $\Psi((\text{Deliver}, \text{Operator}), \sigma_1) = \sigma_1$
 $\Psi((\text{Deliver}, \text{Courier}), \sigma_1) = \sigma_1$

$\Psi((\text{Place}, \text{Operator}), \sigma_2) = \sigma_2$
 $\Psi((\text{Place}, \text{Courier}), \sigma_2) = \sigma_2$
 $\Psi((\text{Pack}, \text{Operator}), \sigma_2) = \sigma_2$
 $\Psi((\text{Pack}, \text{Courier}), \sigma_2) = \sigma_2$
 $\Psi((\text{Ship}, \text{Courier}), \sigma_2) = \sigma_3$
 $\Psi((\text{Ship}, \text{Operator}), \sigma_2) = \sigma_2$
 $\Psi((\text{Deliver}, \text{Operator}), \sigma_2) = \sigma_2$
 $\Psi((\text{Deliver}, \text{Courier}), \sigma_2) = \sigma_2$

$\Psi((\text{Place}, \text{Operator}), \sigma_3) = \sigma_3$
 $\Psi((\text{Place}, \text{Courier}), \sigma_3) = \sigma_3$
 $\Psi((\text{Pack}, \text{Operator}), \sigma_3) = \sigma_3$
 $\Psi((\text{Pack}, \text{Courier}), \sigma_3) = \sigma_3$
 $\Psi((\text{Ship}, \text{Courier}), \sigma_3) = \sigma_3$
 $\Psi((\text{Ship}, \text{Operator}), \sigma_3) = \sigma_3$
 $\Psi((\text{Deliver}, \text{Courier}), \sigma_3) = \sigma_4$
 $\Psi((\text{Deliver}, \text{Operator}), \sigma_3) = \sigma_3$

$\Psi((\text{Place}, \text{Operator}), \sigma_4) = \sigma_4$
 $\Psi((\text{Place}, \text{Courier}), \sigma_4) = \sigma_4$
 $\Psi((\text{Pack}, \text{Operator}), \sigma_4) = \sigma_4$
 $\Psi((\text{Pack}, \text{Courier}), \sigma_4) = \sigma_4$
 $\Psi((\text{Ship}, \text{Operator}), \sigma_4) = \sigma_4$