Description Logics State Features for Planning (dlplan)

Dominik Drexler
Linköping University, Linköping, Sweden
dominik.drexler@liu.se

January 13, 2022

1 Introduction

This document aims at providing a better understanding of the different types of elements and different elements itself that can be used with this library. There are four different types of elements: concepts, roles, boolean, and numericals. The main difference is that each type returns different types of objects when evaluated on a given planning state. Evaluating a concept returns a set of unary relations over objects, evaluating a role returns a set of binary relations over objects, evaluating a boolean returns a Boolean, and evaluating a numerical returns a natural number. The following section shows how elements can be composed to obtain more complex elements.

2 Available Elements

In this section, we describe the elements that are available to construct more complex elements. We make use of description logics [**?**] and include Boolean and numerical elements as described in [**?**].

Consider concepts C, D, roles R, S, T, the universe Δ containing all objects, and X to be either a concept or a role. Also consider some predicate $p(c_0, \ldots, c_{n-1})$ with arity n, integers $0 \leq i, j \leq n-1$, integer $k \in \{0, 1\}$, and some constant a that occurs in every instance.

2.1 Syntax Overview

2.1.1 Concepts

Name	Abstract syntax	Concrete syntax
Primitive concept	$p[i]$	c_primitive(p,i)
Top	$\top$	c_top
Bottom	$\bot$	c_bot
Intersection	$C \sqcap D$	c_and(C, D)
Union	$C \sqcup D$	c_or(C, D)
Negation	$\neg C$	c_not(C)
Difference	$C \setminus D$	c_diff(C, D)
Value restriction	$\forall R.C$	c_all(R, C)
Existential quantification	$\exists R.C$	c_some(R, C)
Role-value-map	$R \subseteq S$	c_subset(R, S)
	$R = S$	c_equal(R, S)
One-of	a	c_one_of(a)
Projection	$R[k]$	c_projection(R, k)

Table 1: Concepts

2.1.2 Roles

Name	Abstract syntax	Concrete syntax
Primitive role	$p[i, j]$	r_primitive(p,i,j)
Universal role	$\top$	r_top
Intersection	$R \sqcap S$	r_and(R, S)
Union	$R \sqcup S$	r_or(R, S)
Complement	$\neg R$	r_not(R)
Difference	$R \setminus S$	r_diff(R, S)
Inverse	R^{-1}	r_inverse(R)
Composition	$R \circ S$	r_compose(R, S)
Transitive closure	R^{+}	r_transitive_closure(R)
Transitive reflexive closure	R^{*}	r_transitive_reflexive_closure(R)
Role restriction	$R\vert_C$	r_restrict(R, C)
Identity	$id(C)$	r_identity(C)

Table 2: Roles

2.1.3 Booleans

Name	Abstract syntax	Concrete syntax
Empty	$\mathit{Empty}(X)$	b_empty(X)
Concept inclusion	$C \sqsubseteq D$	b_inclusion(C, D)
Role inclusion	$R \sqsubseteq S$	b_inclusion(R, S)
Nullary	$\mathit{Nullary}(p)$	b_nullary(p)

Table 3: Booleans

2.1.4 Numericals

Name	Abstract syntax	Concrete syntax
Count	$\mathit{Count}(X)$	n_count(X)
Concept distance	$\mathit{ConceptDistance}(C, R, D)$	n_concept_distance(C, R, D)
Sum concept distance	$\mathit{SumConceptDistance}(C, R, D)$	n_sum_concept_distance(C, R, D)
Role distance	$\mathit{RoleDistance}(R, S, T)$	n_role_distance(R, S, T)
Sum role distance	$\mathit{SumRoleDistance}(R, S, T)$	n_sum_role_distance(R, S, T)

Table 4: Numericals

2.2 Semantics

Our *interpretation* is a state s that consists of ground atoms over a set of predicates.

2.2.1 Concepts

- $(p[i])^s = \{c_i \in \Delta \mid p(c_0, \ldots, c_i, \ldots, c_{arity(p)}) \in s\}$
- $\top^s = \Delta$
- $\bot^s = \emptyset$
- $(C \sqcap D)^s = C^s \cap D^s$
- $(C \sqcup D)^s = C^s \cup D^s$
- $(\neg C)^s = \Delta \setminus C^s$
- $(C \setminus D)^s = (C^s \setminus D^s)$
- $(\forall R.C)^s = \{a \mid \forall b : (a, b) \in R^s \rightarrow b \in C^s\}$
- $(\exists R.C)^s = \{a \mid \exists b : (a, b) \in R^s \wedge b \in C^s\}$
- $(R \subseteq S)^s = \{a \mid \forall b : (a, b) \in R^s \rightarrow (a, b) \in S^s\}$
- $(R = S)^s = \{a \mid \forall b : (a, b) \in R^s \leftrightarrow (a, b) \in S^s\}$

- $a^s = \{a\}$
- $(R[k])^s = \begin{cases} (\exists R.\top)^s & \text{if } k = 0 \\ (\exists R^{-1}.\top)^s & \text{if } k = 1 \end{cases}$

2.2.2 Roles

- $(p[i,j])^s = \{(c_i, c_j) \in \Delta \times \Delta \mid p(c_0, \ldots, c_i, \ldots, c_j, \ldots, c_{arity(p)}) \in s\}$
- $\top^s = \Delta \times \Delta$
- $(R \sqcap S)^s = R^s \cap S^s$
- $(R \sqcup S)^s = R^s \cup S^s$
- $(\neg R)^s = \top^s \setminus R^s$
- $(R \setminus S)^s = (R^s \setminus S^s)$
- $(R^{-1})^s = \{(b,a) \in \Delta \times \Delta \mid (a,b) \in R^s\}$
- $(R \circ S)^s = \{(a,c) \in \Delta \times \Delta \mid (a,b) \in R^s \wedge (b,c) \in S^s\}$
- $(R^+)^s = \bigcup_{n \geq 1} (R^s)^n$
- $(R^*)^s = \bigcup_{n \geq 0} (R^s)^n$
- $(R|_C)^s = R^s \sqcap (\Delta \times C^s)$
- $(id(C))^s = \{(d,d) \mid d \in C^s\}$

where the iterated composition $(R^s)^n$ is inductively defined as $(R^s)^0 = \{(d,d) \mid d \in \Delta\}$ and $(R^s)^{n+1} = (R^s)^n \circ R^s$.

2.2.3 Booleans

- $\mathit{Empty}(X)^s$ is true iff $|X^s| = 0$
- $(C \sqsubseteq D)^s$ is true iff $C^s \subseteq D^s$
- $(R \sqsubseteq S)^s$ is true iff $R^s \subseteq S^s$
- $\mathit{Nullary}(p)^s$ is true iff $p() \in s$

2.2.4 Numericals

- *Count*$(X)^s \equiv |X^s|$
- *ConceptDistance*$(C, R, D)^s$ is the smallest $n \in \mathbb{N}_0$ s.t. there are objects $x_0, \ldots, x_n$ with $x_0 \in C^s$, $x_n \in D^s$ and $(x_i, x_{i+1}) \in R^s$ for $i = 0, \ldots, n-1$. Special cases: If C^s is empty then the element evaluates to 0 and if no such n exists then it evaluates to ∞.
- *SumConceptDistance*$(C, R, D)^s := \sum_{x \in C^s}$ *ConceptDistance*$(\{x\}, R, D)^s$ where the sum evaluates to ∞ if any term is ∞.
- *RoleDistance*$(R, S, T)^s$ is the smallest $n \in \mathbb{N}_0$ s.t. there are objects $x_0, \ldots, x_n$, there exists some $(a, x_0) \in R^s$, $(a, x_n) \in T^s$, and $(x_i, x_{i+1}) \in S^s$ for $i = 0, \ldots, n-1$. Special cases: If R^s is emtpy then the element evaluates to 0 and if no such n exists then it evaluates to ∞.
- *SumRoleDistance*$(R, S, T)^s := \sum_{r \in R^s}$ *RoleDistance*$(\{r\}, S, T)^s$, where the sum evaluates to ∞ if any term is ∞.

3 Feature Generation

In the feature generation, we place the following additional restrictions on the above elements in order to decrease the combinatorial blowup.

- r_restrict(R, C), r_inverse(R), r_transitive_closure(R), c_equal(R, S), r_transitive_reflexive_closure(R) where R, S are a primitive roles and C is a primitive concept
- *ConceptDistance*(C, R, D) where R is restricted to complexity at most 2