Comparison of STRIPS, ADL and PDDL

Explanation of PDDL

We can use specification languages like STRIPS or ADL to describe a system. Another possibility to describe systems is the Planning Domain Definition Language (PDDL). It primarily describes a system using a set of preconditions and post-conditions.

PDDL is a domain definition language which is supported by most planners. It is used to define the properties of a domain, the predicates which are used and the action definition.

A predicate defines the property of an object which can be true or false, e.g. yellow t-shirt. Yellow is the property and t-shirt is the object.

- Like in every other programming language you can define variables and types, e.g.

  ?iterator - int
  ?ms - myStructure

- Representation of and, not and or:

  (and (Yellow T-shirt)
       (Big Shoes))
  (not (Yellow Shoes))
  (or (Green Shoes)
       (not (Yellow T-shirt)))

- Like STRIPS and contrary to ADL the PDDL Language uses the closed world assumption.
- Nevertheless the forall operator is usefull.

E.g. let all the t-shirts in the world be yellow:

  (:types T-shirt)
  (:predicates(Yellow ?things – T-shirt))
  ...
  (forall(?things – T-shirt)(Yellow ?things))

This implies that the property yellow should be true for all objects in the domain that are of type T-shirt.

- Also known from the mathematical set theory: the exists operator:

E.g. there exists even one green shoe:

  (:types Shoes)
  (:predicates(Green ?things – Shoes))
  ...
  (exists(?things – Shoes)(Green Shoes ?things))
This evaluates TRUE if there exists one or more objects which has the property green.

**PDDL – Domain Definition, Problem Definition**

To define a domain with PDDL you have to define
- Requirements to declare which packages are used.
- Types to define own types.
- Constants.
- Predicates to define the truth statement.
- Action operators with a precondition (predicates have to be true before the operator applies) and effects (predicates become true after the operator is applied).

To define a problem you have to define an initial (predicates which are true at the beginning of the problem) and a goal state (predicates which are true at the end of the problem).

**Example – the air cargo transport problem**

Now the three languages are explained with one example, the air cargo transport problem: This problem is involving loading and unloading cargo onto and off planes and flying it from place to place. The problem can be defined with three actions: Load, unload and fly.

**STRIPS**

The complete air cargo transport example:

**3 actions:** Load, Unload, Fly

variables: a: Airport, c: Cargo, p: Plane, from: start, to: destination
possible substitutions for a, from, to: JFK – John F. Kennedy Airport New York, SFO – San Francisco International Airport
c: C1, C2
p: P1, P2

Action **Load** (c, p, a),
PRECOND: At (c, a) ∧ At (p, a) ∧ Cargo (c) ∧ Plane (p) ∧ airport (a)
EFFECT: ¬At (c, a) ∧ In (c, p))

Action **Unload** (c, p, a),
PRECOND: In (c, p) ∧ At (p, a) ∧ Cargo (c) ∧ Plane (p) ∧ Airport (a)
EFFECT: At (c, a) ∧ ¬In (c, p))

Action **Fly** (p, from, to),
PRECOND: At (p, from) ∧ Plane (p) ∧ Airport (from) ∧ Airport (to)
EFFECT: ¬At (p, from) ∧ At (p, to))

**Initial State:**
Init (At (C1, SFO) ∧ At (C2, JFK) ∧ At (P1, SFO) ∧ At (P2, JFK) ∧ Cargo (C1) ∧ Cargo (C2) ∧ Plane (P1) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO))

**Goal State:**
Goal (At (C1, JFK) ∧ At (C2, SFO))
Init (At (C1, SFO) ∧ At (C2, JFK) ∧ At (P1, SFO) ∧ At (P2, JFK) ∧ Cargo (C1) ∧ Cargo (C2) ∧ Plane (P1) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO))

- **First Action: Load (C1, P1, SFO)**

Action (Load (C1, P1, SFO),
    PRECOND: At (C1, SFO) ∧ At (P1, SFO) ∧ Cargo (C1) ∧ Plane (P1) ∧ airport (SFO)
    EFFECT: ¬At (C1, SFO) ∧ In (C1, P1))

S1 (In (C1, P1) ∧ At (C2, JFK) ∧ At (P1, SFO) ∧ At (P2, JFK) ∧ Cargo (C1) ∧ Cargo (C2) ∧ Plane (P1) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO))

- **Second Action: Fly (P1, SFO, JFK)**

Action (Fly (P1, SFO, JFK),
    PRECOND: At (P1, SFO) ∧ Plane (P1) ∧ Airport (SFO) ∧ Airport (JFK)
    EFFECT: ¬At (P1, SFO) ∧ At (P1, JFK))

S2 (In (C1, P1) ∧ At (C2, JFK) ∧ At (P1, JFK) ∧ At (P2, JFK) ∧ Cargo (C1) ∧ Cargo (C2) ∧ Plane (P1) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO))

- **Third Action: Load (C2, P2, JFK)**

Action (Load (C2, P2, JFK),
    PRECOND: At (C2, JFK) ∧ At (P2, JFK) ∧ Cargo (C2) ∧ Plane (P2) ∧ airport (JFK)
    EFFECT: ¬At (C2, JFK) ∧ In (C2, P2))

S3 (In (C1, P1) ∧ In (C2, P2) ∧ At (P1, JFK) ∧ At (P2, JFK) ∧ Cargo (C1) ∧ Cargo (C2) ∧ Plane (P1) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO))

- **Fourth Action: Fly (P2, JFK, SFO)**

Action (Fly (P2, JFK, SFO),
    PRECOND: At (P2, JFK) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO)
    EFFECT: ¬At (P2, JFK) ∧ At (P2, SFO))

S4 (In (C1, P1) ∧ In (C2, P2) ∧ At (P1, JFK) ∧ At (P2, SFO) ∧ Cargo (C1) ∧ Cargo (C2) ∧ Plane (P1) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO))

- **Fifth Action: Unload (C1, P1, JFK)**

Action (Unload (C1, P1, JFK),
    PRECOND: In (C1, P1) ∧ At (P1, JFK) ∧ Cargo (C1) ∧ Plane (P1) ∧ Airport (JFK)
    EFFECT: At (C1, JFK) ∧ ¬In (C1, P1))

S5 (At (C1, JFK) ∧ In (C2, P2) ∧ At (P1, JFK) ∧ At (P2, SFO) ∧ Cargo (C1) ∧ Cargo (C2) ∧ Plane (P1) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO))
• **Sixth Action: Unload** (C2, P2, SFO)

Action *(Unload* (C2, P2, SFO),
  
  **PRECOND**: In (C2, P2) ∧ At (P2, SFO) ∧ Cargo (C2) ∧ Plane (P2) ∧ Airport (SFO)
  
  **EFFECT**: At (C2, SFO) ∧ ¬In (C2, P2))

S6 (At (C1, JFK) ∧ At (C2, SFO) ∧ At (P1, JFK) ∧ At (P2, SFO) ∧ Cargo (C1) ∧ Cargo (C2) ∧ Plane (P1) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO))

Goal (At (C1, JFK) ∧ At (C2, SFO))

A state *s* **satisfies** a goal *g* if *s* contains all the atoms in *g* (and possibly others)

  S6 satisfies Goal
ADL

The complete air cargo transport example:

3 actions: Load, Unload, Fly

Action (Load (c: cargo, p: plane, a: airport),
  PRECOND: At (c, a) \land At (p, a)
  EFFECT: \lnot At (c, a) \land \lnot In (c, p))

Action (Unload (c: cargo, p: plane, a: airport),
  PRECOND: In (c, p) \land At (p, a)
  EFFECT: At (c, a) \land \lnot In (c, p))

Action (Fly (p: plane, from: airport, to: airport),
  PRECOND: At (p, from) \land (from \neq to)
  EFFECT: \lnot At (p, from) \land At (p, to))

Initial State:
Init (At (C1, SFO) \land At (C2, JFK) \land At (P1, SFO) \land At (P2, JFK) \land (C1:Cargo) \land (C2:Cargo)
  \land (P1:Plane) \land (P2:Plane) \land (JFK:Airport) \land (SFO:Airport) \land (SFO \neq JFK))

Goal State:
Goal (At (C1, JFK) \land At (C2, SFO))

- First Action: Load (C1, P1, SFO)

Action (Load (C1, P1, SFO),
  PRECOND: At (C1, SFO) \land At (P1, SFO)
  EFFECT: \lnot At (C1, SFO) \land In (C1, P1))

S1 (\lnot At (C1, SFO) \land In (C1, P1) \land At (C2, JFK) \land At (P1, SFO) \land At (P2, JFK) \land (C1:Cargo) \land (C2:Cargo) \land (P1:Plane) \land (P2:Plane) \land (JFK:Airport) \land (SFO:Airport) \land (SFO \neq JFK))

- Second Action: Fly (P1,SFO,JFK)

Action (Fly (P1, SFO, JFK),
  PRECOND: At (P1, SFO) \land (SFO \neq JFK)
  EFFECT: \lnot At (P1, SFO) \land At (P1, JFK))

S2 (\lnot At (P1, SFO) \land At (P1, JFK) \land \lnot At (C1, SFO) \land In (C1, P1) \land At (C2, JFK) \land At (P2, JFK) \land (C1:Cargo) \land (C2:Cargo) \land (P1:Plane) \land (P2:Plane) \land (JFK:Airport) \land (SFO:Airport) \land (SFO \neq JFK))
• Third Action: Load (C2, P2, JFK)

Action (Load (C2, P2, JFK),
PRECOND: At (C2, JFK) \& At (P2, JFK)
EFFECT: \neg At (C2, JFK) \& In (C2, P2))

S3 (\neg At (C2, JFK) \& In (C2, P2) \& \neg At (P1, SFO) \& At (P1, JFK) \& \neg At (C1, SFO) \& In (C1, P1) \& At (P2, JFK) \& (C1:Cargo) \& (C2:Cargo) \& (P1:Plane) \& (P2:Plane) \& (JFK:Airport) \& (SFO:Airport) \& (SFO \neq JFK))

• Fourth Action: Fly (P2,JFK,SFO)

Action (Fly (P2, JFK, SFO),
PRECOND: At (P2, JFK) \& (SFO \neq JFK)
EFFECT: \neg At (P2, JFK) \& At(P2, SFO))

S4 (\neg At (P2, JFK) \& At(P2, SFO) \& \neg At (C2, JFK) \& In (C2, P2) \& \neg At (P1, SFO) \& At (P1, JFK) \& \neg At (C1, SFO) \& In (C1, P1) \& (C1:Cargo) \& (C2:Cargo) \& (P1:Plane) \& (P2:Plane) \& (JFK:Airport) \& (SFO:Airport) \& (SFO \neq JFK))

• Fifth Action: Unload (C1, P1, JFK)

Action (Unload (C1, P1, JFK),
PRECOND: In (C1, P1) \& At (P1, JFK)
EFFECT: At (C1, JFK) \& \neg In (C1, P1))

S5 (At (C1, JFK) \& \neg In (C1, P1) \& \neg At (P2, JFK) \& At(P2, SFO) \& \neg At (C2, JFK) \& In (C2, P2) \& \neg At (P1, SFO) \& At (P1, JFK) \& \neg At (C1, SFO) \& (C1:Cargo) \& (C2:Cargo) \& (P1:Plane) \& (P2:Plane) \& (JFK:Airport) \& (SFO:Airport) \& (SFO \neq JFK))

• Sixth Action: Unload (C2, P2, SFO)

Action (Unload (C2, P2, SFO),
PRECOND: In (C2, P2) \& At (P2, SFO)
EFFECT: At (C2, SFO) \& \neg In (C2, P2))

S6 (At (C2, SFO) \& \neg In (C2, P2) \& At (C1, JFK) \& \neg In (C1, P1) \& \neg At (P2, JFK) \& At(P2, SFO) \& \neg At (C2, JFK) \& \neg At (P1, SFO) \& At (P1, JFK) \& \neg At (C1, SFO) \& (C1:Cargo) \& (C2:Cargo) \& (P1:Plane) \& (P2:Plane) \& (JFK:Airport) \& (SFO:Airport) \& (SFO \neq JFK))

Goal (At (C1, JFK) \& At (C2, SFO))

A state \(s\) satisfies a goal \(g\) if \(s\) contains all the atoms in \(g\) (and possibly others)

\(\rightarrow\) S6 satisfies Goal
PDDL

(define (domain air-cargo)
  (:requirements :typing :adl)
  (:types cargo plane airport)
  (:predicates (at ?t - (either cargo plane) ?a - airport)
               (in ?c - cargo ?p - plane))

  (:action load
   :parameters (?c - cargo ?p - plane ?a - airport)
   :precondition (and (at ?c ?a) (at ?p ?a))
   :effect (and (not (at ?c ?a)) (in ?c ?p)))

  (:action unload
   :parameters (?c -cargo ?p - plane ?a - airport)
   :effect (and (at ?c ?a) (not (in ?c ?p))))

  (:action fly
   :parameters (?p - plane ?a1 ?a2 - airport)
   :precondition (and (at ?p ?a1) (not (= ?a1 ?a2)))
   :effect (and (not (at ?p ?a1)) (at ?p ?a2)))
)

(define (problem sfo-jfk)
  (:domain air-cargo)
  (:objects c1 c2 - cargo sfo jfk - airport p1 p2 - plane)
  (:init (at c1 sfo)
         (at p1 sfo)
         (at c2 jfk)
         (at p2 jfk)
         (at c1 sfo)
         (at c2 jfk)
  (:goal (and (at c1 jfk) (at c2 sfo)))
)
**Main Sources:**

Stuart Russell, Peter Norvig: Artificial Intelligence, New Jersey, 2003  
http://www.informatik.uni-freiburg.de/~koehler/aips/PDDL-MANUAL.ps.gz  
http://www.ksl.stanford.edu/people/fikes/cs222/2001/pddl.ps  
http://www.ai.sri.com  
http://www.cs.washington.edu/homes/weld  
http://www.2.cs.cmu.edu/~aips98/