Research Review

Planning Search

Version 2

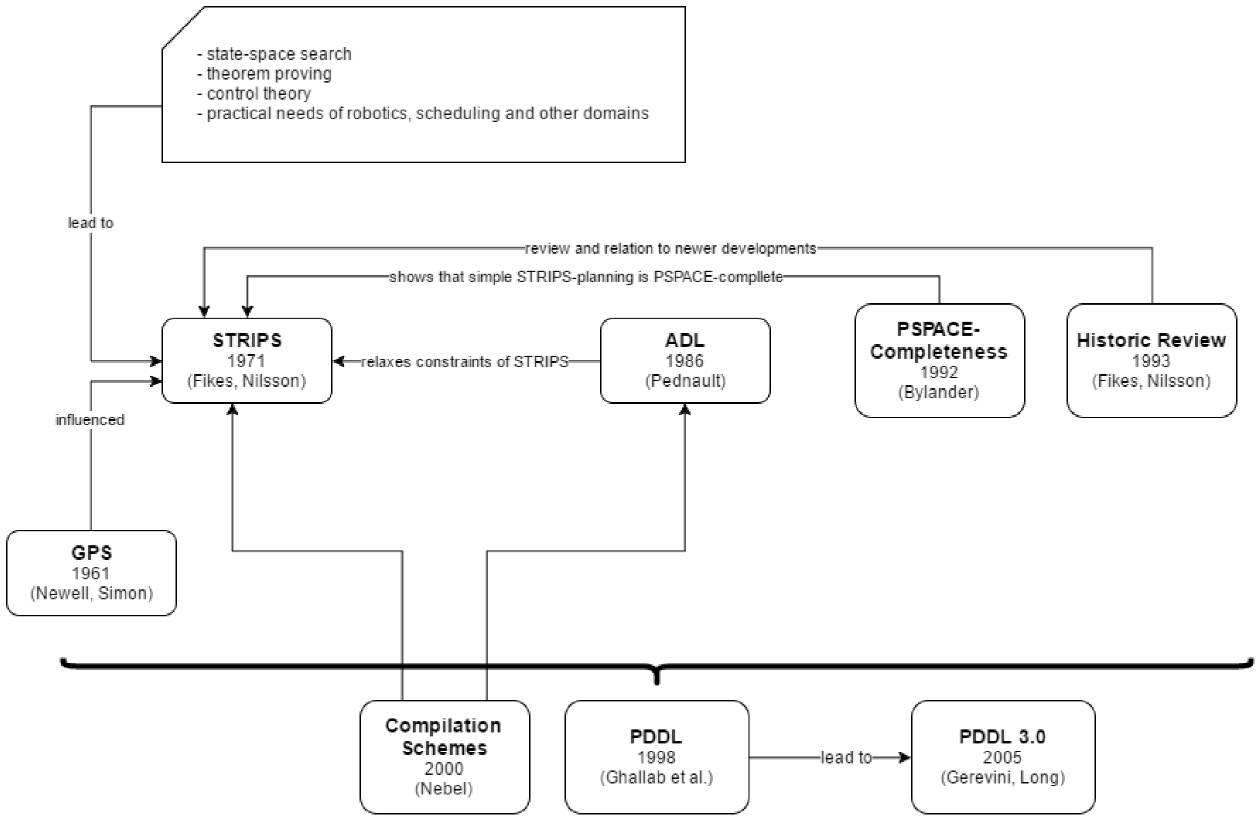
Andreas Mayer

27th February, 2017

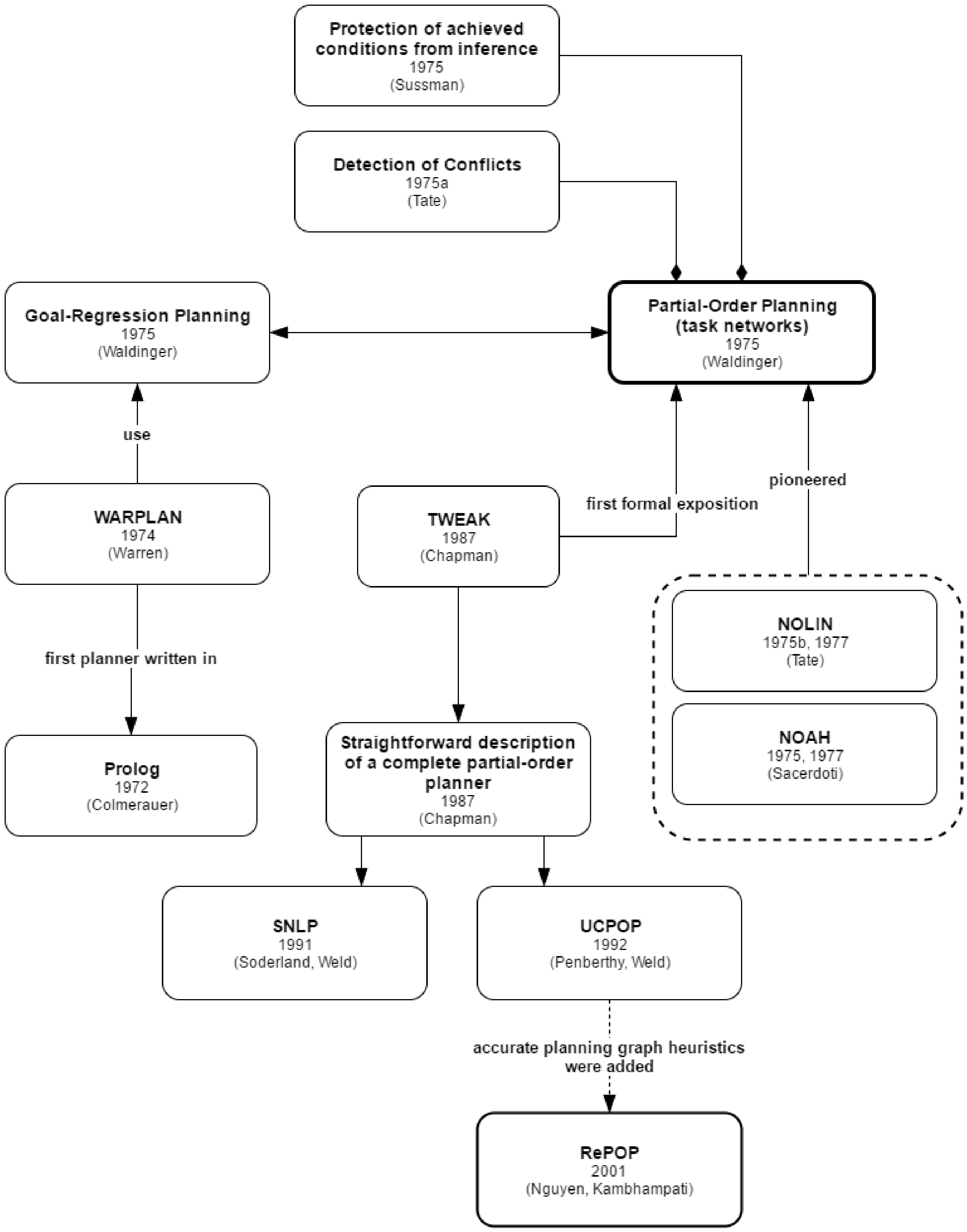
Overview

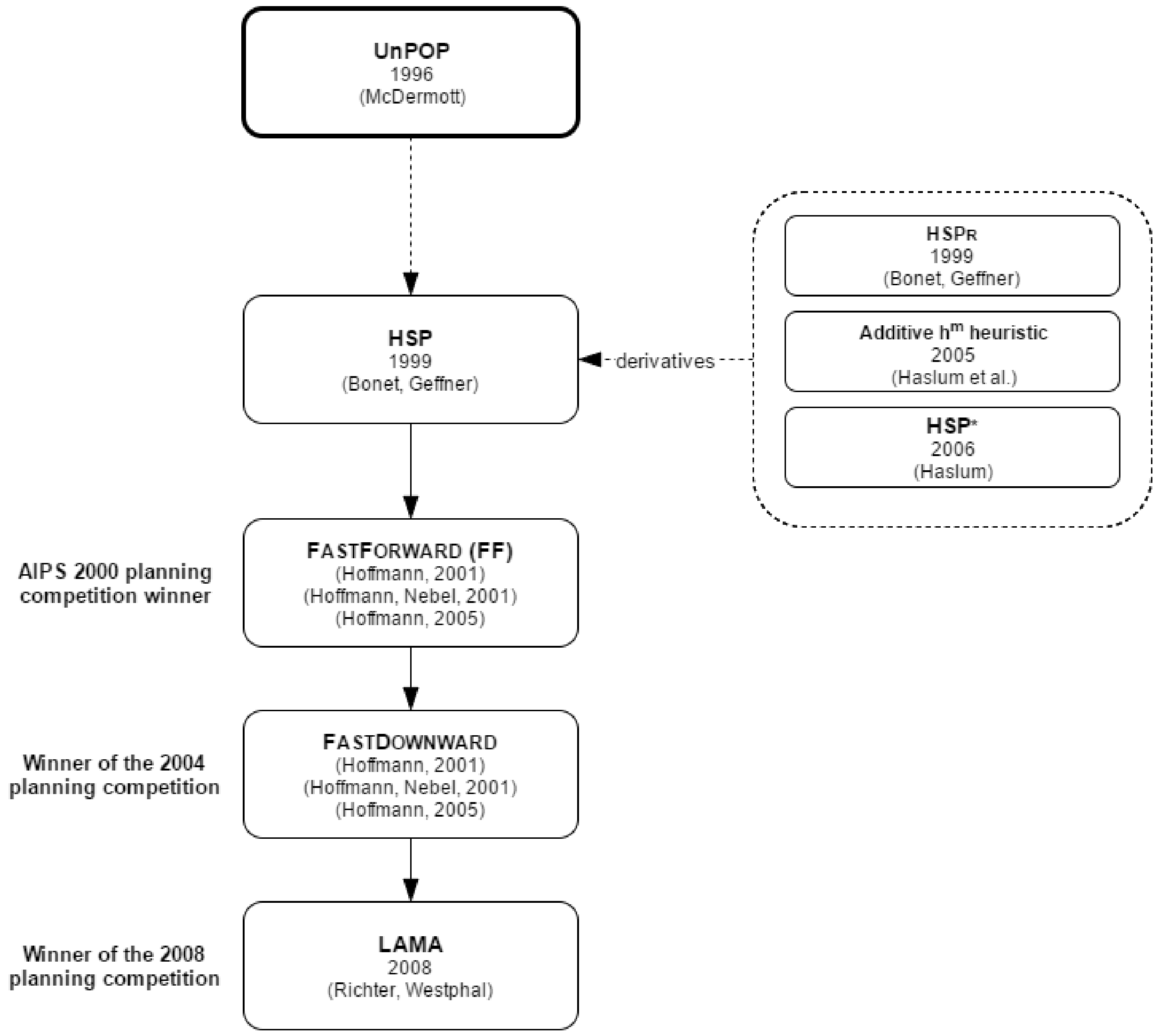
"AI planning arose from investigations into state-space search, theorem proving, and control theory and from the practical needs of robotics, scheduling, and other domains." (AIMA, Stuart Russel and Peter Norvig, 2010)

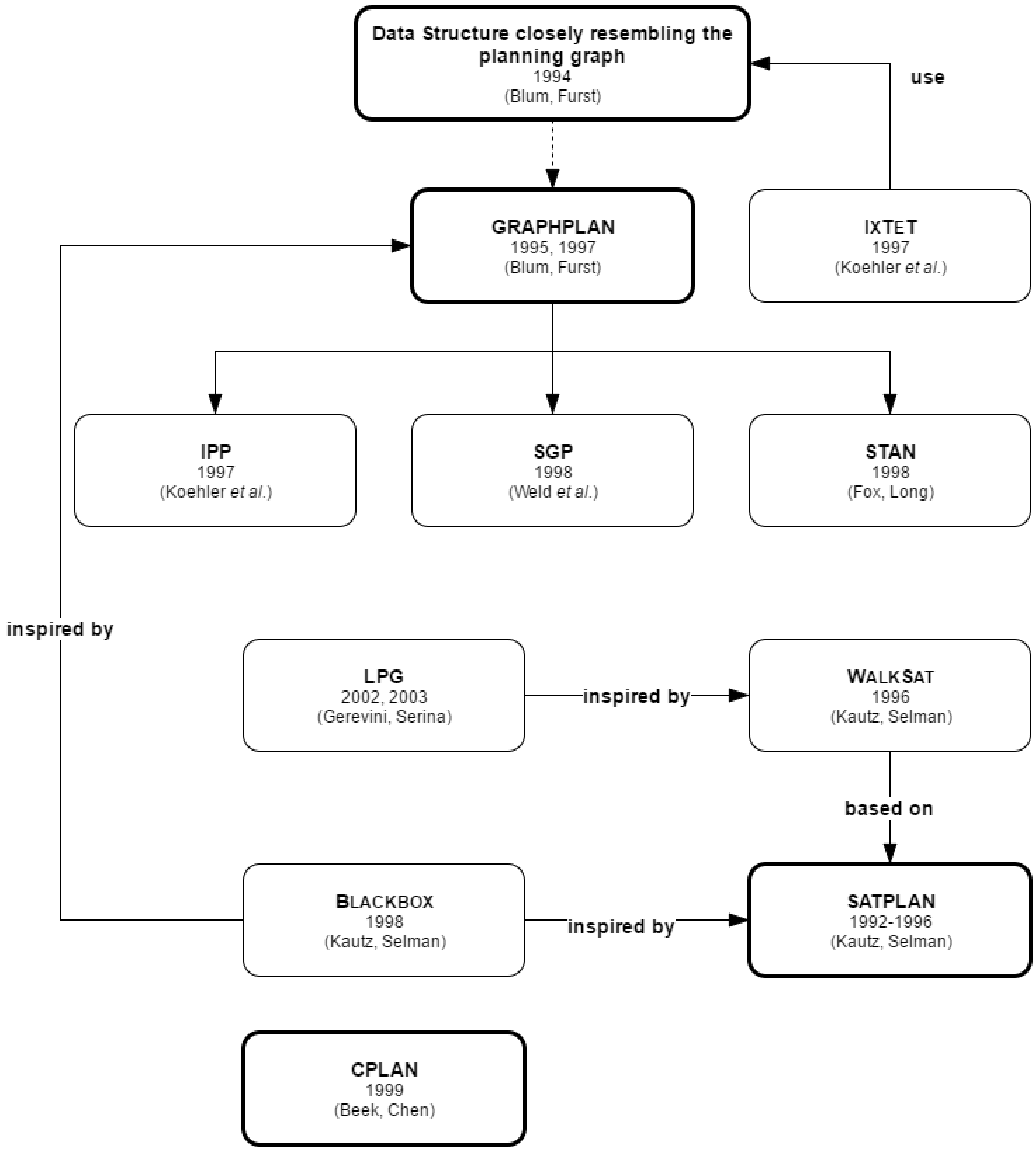
The following images illustrate the the relationship between several of the contributions to the AI planning domain. The components are based on the bibliographical and historical notes from chapter 10 of the AIMA book.



- state-space search
- theorem proving
- control theory
- practical needs of robotics, scheduling and other domains
lead to
review and relation to newer developments
shows that simple STRIPS-planning is PSPACE-compllete
STRIPS 1971 (Fikes, Nilsson)
relaxes constraints of STRIPS
ADL 1986 (Pednault)
PSPACE-Completeness 1992 (Bylander)
Historic Review 1993 (Fikes, Nilsson)
influenced
GPS 1961 (Newell, Simon)
Compilation Schemes 2000 (Nebel)
PDDL 1998 (Ghallab et al.)
lead to
PDDL 3.0 2005 (Gerevini, Long)


Protection of achieved conditions from inference
1975
(Sussman)
Detection of Conflicts
1975a
(Tate)
Goal-Regression Planning
1975
(Waldinger)
Partial-Order Planning (task networks)
1975
(Waldinger)
use
WARPLAN
1974
(Warren)
TWEAK
1987
(Chapman)
first formal exposition
pioneered
NOLIN
1975b, 1977
(Tate)
NOAH
1975, 1977
(Sacerdoti)
first planner written in
Prolog
1972
(Colmerauer)
Straightforward description of a complete partial-order planner
1987
(Chapman)
SNLP
1991
(Soderland, Weld)
UCPOP
1992
(Penberthy, Weld)
accurate planning graph heuristics were added
RePOP
2001
(Nguyen, Kambhampati)


UnPOP
1996
(McDermott)
HSP
1999
(Bonet, Geffner)
derivatives
HSPR
1999
(Bonet, Geffner)
Additive h^m heuristic
2005
(Haslum et al.)
HSP*
2006
(Haslum)
AIPS 2000 planning competition winner
FASTFORWARD (FF)
(Hoffmann, 2001)
(Hoffmann, Nebel, 2001)
(Hoffmann, 2005)
Winner of the 2004 planning competition
FASTDOWNWARD
(Hoffmann, 2001)
(Hoffmann, Nebel, 2001)
(Hoffmann, 2005)
Winner of the 2008 planning competition
LAMA
2008
(Richter, Westphal)


Data Structure closely resembling the planning graph
1994
(Blum, Furst)
use
GRAPHPLAN
1995, 1997
(Blum, Furst)
IXTET
1997
(Koehler et al.)
IPP
1997
(Koehler et al.)
SGP
1998
(Weld et al.)
STAN
1998
(Fox, Long)
inspired by
LPG
2002, 2003
(Gerevini, Serina)
inspired by
WALKSAT
1996
(Kautz, Selman)
based on
BLACKBOX
1998
(Kautz, Selman)
inspired by
SATPLAN
1992-1996
(Kautz, Selman)
CPLAN
1999
(Beek, Chen)

Review Introduction

All of the three languages that are described in this review are subsets of First-Order-Propositional-Logic but they offer different degrees of expressiveness. A search problem described in a planning language usually consists of the following components.

- *State space*
- *Initial state* of the world
- *Goals* to be reached
- Operators that can be executed to change the state of the world

Operators consist of the following elements:

- *Action*
- *Precondition*
- *Effect*

Coming up with good operators is not a trivial task due to the following problems[1,2]:

- Qualification Problem
- Ramification Problem
- Frame Problem

[1] Hendler, Proceedings of the First Conference (AIPS 92)
[2] Ghallab, Nao, Traverso (2004)

Review

STRIPS

STanford **R**esearch **I**nstitute **P**roblem **S**olver, or **STRIPS** was developed at the Stanford Research Institute to solve problems by searching a space of "world models" to find a state that complies to a given number of goals. STRIPS was initially developed to tackle problems from the robotics domain[3] which have a higher degree of complexity than simple puzzles or block games. (Richard E. Fikes, Nils J. Nilsson, Winter 1971)

The STRIPS framework was a central component in the domain of problem solving and has lead to a log of research in artificial intelligence. (Nilsson, 1991)

ADL

The **A**ction **D**efinition **L**anguage, or ADL (Pednault, 1986) is based on STRIPS but it relaxes some of the constraints that STRIPS imposes and "*made it possible to encode more realistic problems*." (Stuart Russel and Peter Norvig, 2010)

For example it is possible to use *negations* and *disjunctions* in ADL. Other extensions to the language can be found in the Comparison chapter.

PDDL

Planning **D**omain **D**efinition **L**anguage, or **PDDL** (Ghallab *et al.*, 1998) is a "*computer-parsable, standardized syntax"* (Russel, Norvig, 2010) that was introduced as the standard language for the International Planning Competition since 1998. It was inspired by STRIPS and ADR among others. Over the years several extensions were implemented to improve the expressiveness of the language. The latest version is *PDDL 3.1.*

I have chosen **STRIPS**, **ADL** and **PDDL** for my review as the fact that PDDL is still used today[4], clearly states its importance and the power behind this research result. PDDL took several evolutionary steps and there exist successors, variant and extensions of PDDL[5].

[3] It was developed to control the robot "Shakey".

[4] See http://icaps17.icaps-conference.org/workshops/KEPS/ as an example.

[5] https://en.wikipedia.org/wiki/Planning_Domain_Definition_Language

Comparison

The following non-exhaustive table illustrates some differences between the three languages. Green field indicate that the given expression/methodology is supported, red fields indicate that it is not supported.

In the **closed-world** assumption all unmentioned literals are being considered as false, in the **open-world** assumption they are unknown.

	STRIPS	**ADL**	**PDDL**
World Assumption	Closed-World	Open-World	Closed-World
Positive Literals	🟩 **A**	🟩 **A**	🟩 **A**
Conjunctions	🟩 **A ⋀ B ⋀ C**	🟩 **A ⋀ B ⋀ C**	🟩 **A ⋀ B ⋀ C**
Disjunctions in Goals	🟥	🟩 (A ⋀ (**B ⋁ C**))	🟩 (A ⋀ (**B ⋁ C**))
Negative Literals	🟥	🟩 **¬A ⋀ ¬B**	🟩 **¬A ⋀ ¬B**
Quantifiable variables in goas	🟥	🟩 **∃x At (P1, x) ⋀ At(P2, x)**	🟩 **∃x At (P1, x) ⋀ At(P2, x)**
Equality	🟥	🟩 **x = y**	🟩 **x = y**
Types	🟥	🟩 **a : Airplane**	🟩 **a : Airplane**
Preferences & State-trajectory constraints	🟥	🟥	🟩 **PDDL 3.0**[6]
Object-fluents	🟥	🟥	🟩 **PDDL 3.1**[7]

[6] Gerevini, A.; Long, D. (2005)
[7] Helmert, M. (2008) - *Unpublished summary from the IPC-2008 website*.

Examples

This chapter includes examples of the three languages selected for the review. (Martin Kalisch and Stefan König, 2005).

STRIPS

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))`

ADL

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

Goal State:
Goal (At (C1, JFK) ∧ At (C2, SFO))

PDDL

```
(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)
      (:goal
            (and (at c1 jfk) (at c2 sfo))
      )
)
```