

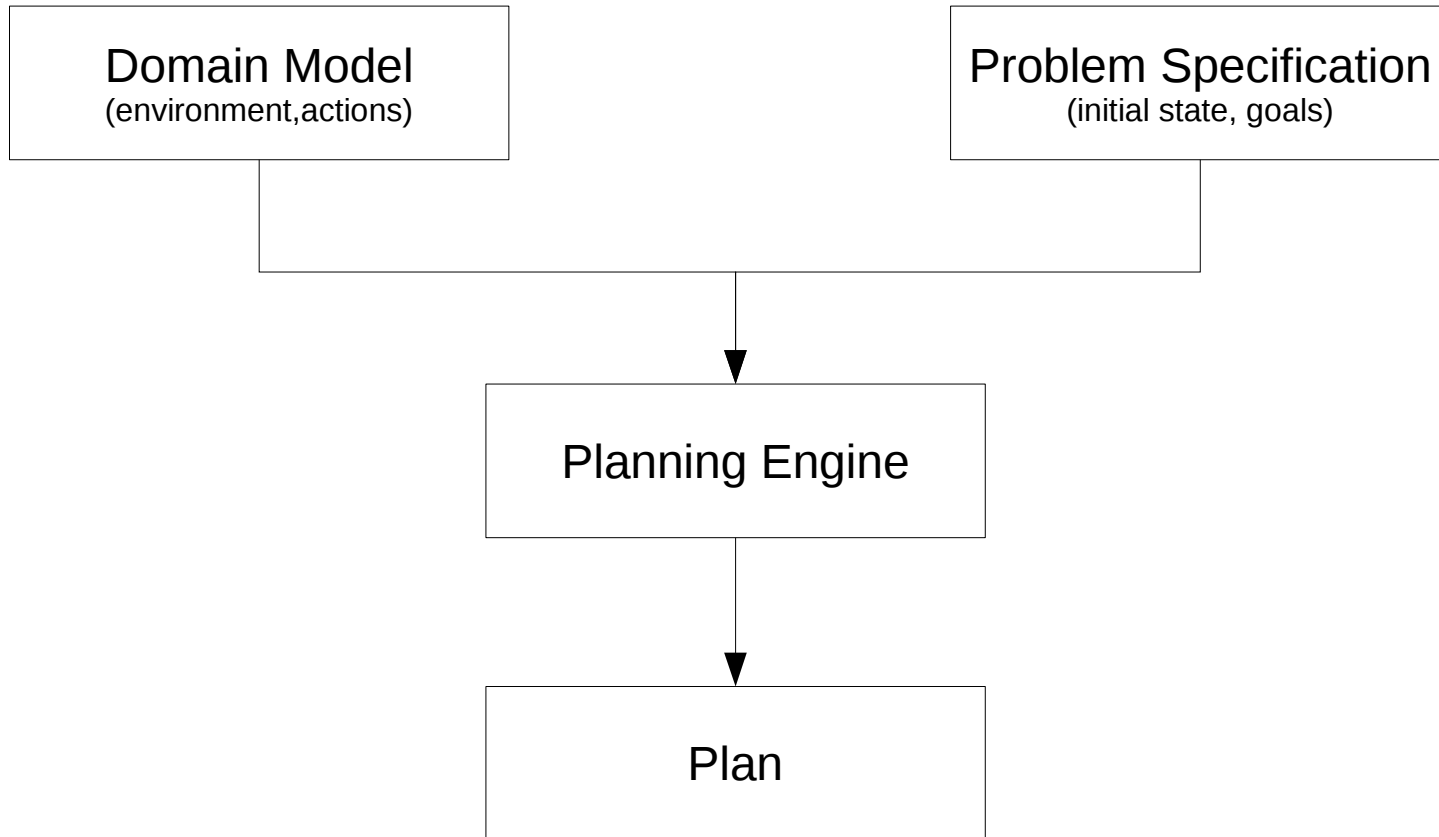


Planning for AI (B4M36PUI)

Modelling Languages, Knowledge
Engineering Tools, Domain Reformulation

Lukáš Chrpa

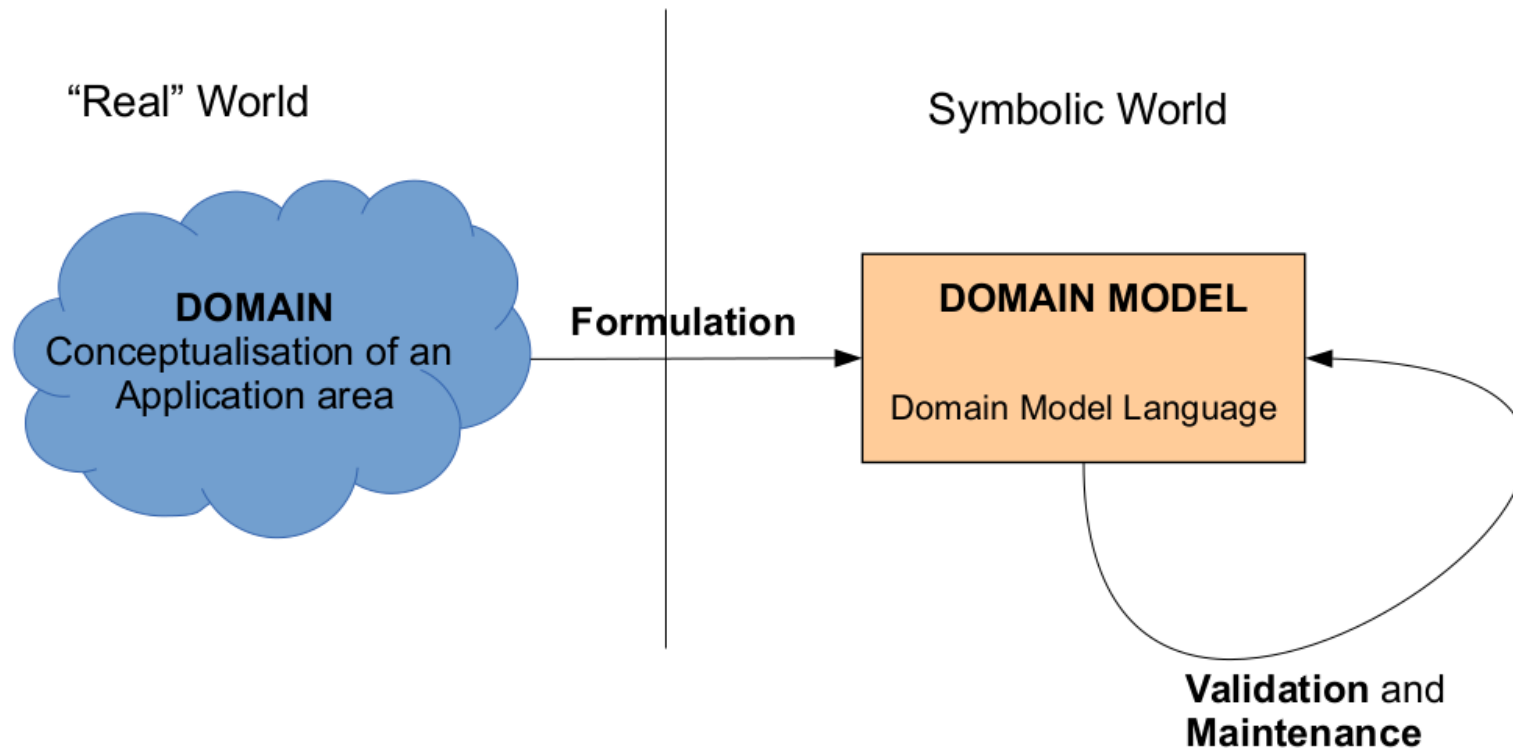
Domain-independent Planning Concept



Domain-independent Planning Concept

- A (description) **language**
 - Describe domain model and problem specification
(usually one domain model for a class of problems)
- A **planning engine**
 - must support the language
 - should be efficient for the given domain model
- **Plans** interpreting

Knowledge Engineering in Planning

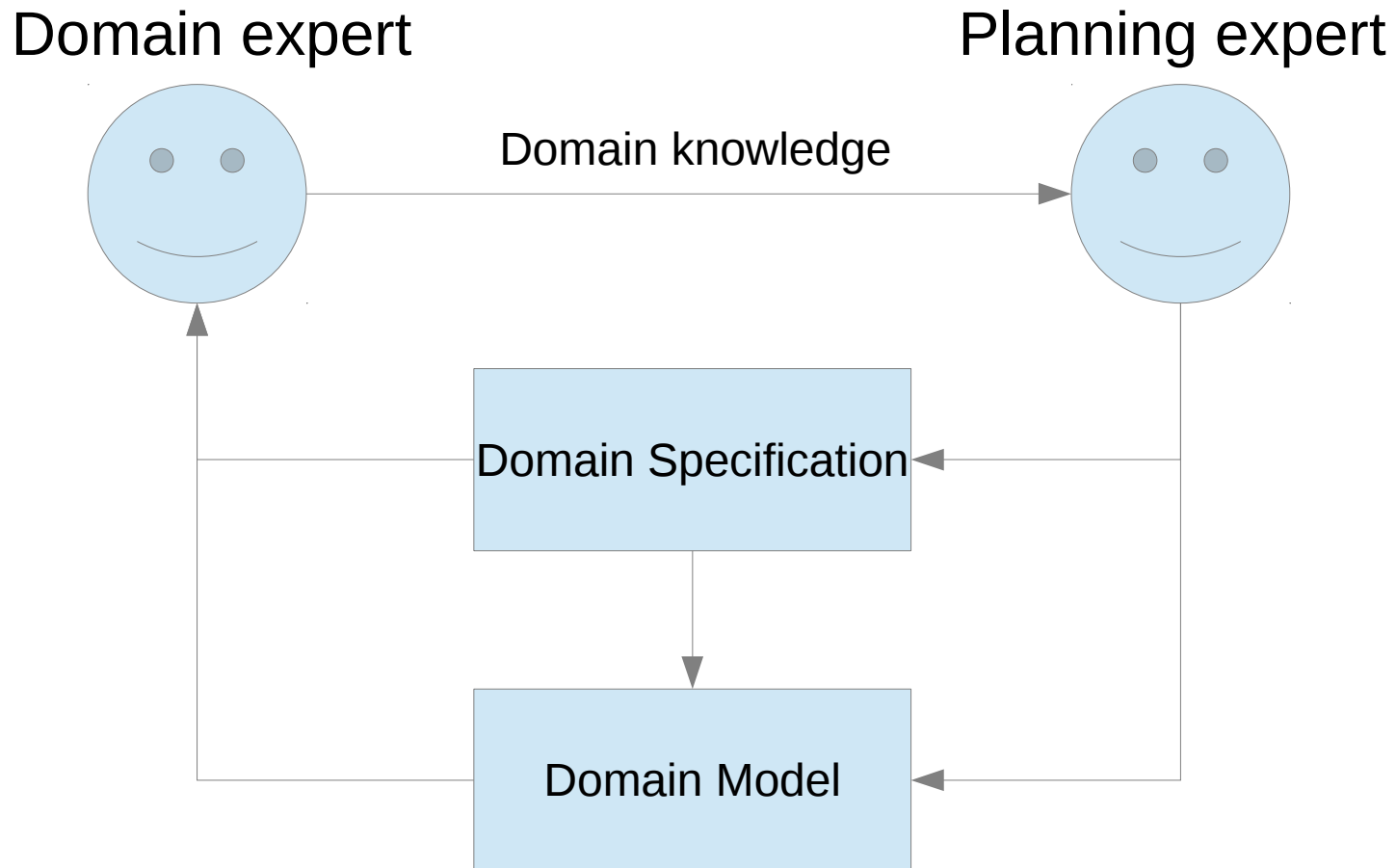




Properties of a Domain Model

- **Accuracy** – There is a mapping between domain requirements and a domain model
- **Consistency** – All assertions (invariants) are true
- **Completeness** – Solution plans correspond to real-world solutions
- **Adequacy** – A domain model is expressive enough to capture domain requirements
- **Operationality** – Planning engines can find solution plans in reasonable time/memory constraints

Knowledge Engineering Process



An iterative process – can take a long time !



Modelling Languages

PDDL [McDermott et al, 1998]

- Planning Domain Definition Language (PDDL)
- Inspired by the STRIPS and ADL languages
- Most widespread
- Official language of International Planning Competitions (IPCs)

```
(define (domain blocksworld)
  (:requirements :strips :typing)
  (:types block)
  (:predicates (on ?x - block ?y - block)
    (ontable ?x - block)
    (clear ?x - block)
    (handempty)
    (holding ?x - block)
  )
  (:action pick-up
    :parameters (?x - block)
    :precondition (and (clear ?x)
      (ontable ?x)
      (handempty))
    :effect (and (not (ontable ?x))
      (not (clear ?x))
      (not (handempty))
      (holding ?x))
  )
  ...
)
```




Versions of PDDL

- **PDDL 1.2**
 - Predicate centric (i.e., classical representation)
 - Object types
 - ADL features (e.g., conditional effects, equality)
- **PDDL 2.1**
 - Numeric Fluents
 - Durative Actions
- **PDDL 2.2**
 - Timed-initial literals
 - Derived Predicates
- **PDDL 3.0**
 - State-trajectory constraints (hard constraints for the planning process)
 - Preferences (soft constraints for the planning process)
- **PDDL 3.1**
 - Object Fluents



Extensions of PDDL

- **PDDL+**
 - Continuous processes
 - Exogenous events
- **PPDDL**
 - Probabilistic action effects
 - Reward fluents
- **MA-PDDL**
 - Multi-agent planning

NDDL [Frank & Jonsson, 2002]

- NASA's response to PDDL
- Variable representation
- Timelines/activities
- Constraints between activities

```
class Instrument
{
    Rover rover;
    InstrumentLocation location;
    InstrumentState state;

    Instrument(Rover r)
    {
        rover = r;
        location = new InstrumentLocation();
        state = new InstrumentState();
    }

    action TakeSample{
        Location rock;
        eq(10, duration);
    }
    ...
}

Instrument::TakeSample
{
    met_by(condition object.state.Placed on);
    eq(on.rock, rock);

    contained_by(condition object.location.Unstowed);

    equals(effect object.state.Sampling sample);
    eq(sample.rock, rock);

    starts(effect object.rover.mainBattery.consume tx);
    eq(tx.quantity, 120); // consume battery power
}
```

<https://github.com/nasa/europa/wiki/Example-Rover>

ANML [Smith et al., 2008]

- Combines aspects from NDDL and PDDL
 - Actions and states (PDDL)
 - Variable representation (NDDL)
 - Temporal Constraints (NDDL)
- Hierarchical methods

```
action Pickup (crew ev, object item)
{
  duration := 5 ;
  [start] located(ev) == located(item);
  [all] possesses(ev,item) == FALSE:
  ->TRUE ;
  [end] located(item) := POSSESSED ;
}
```

```
action Putaway (crew ev, object item,
location stowage)
{
  Duration := 10 ;
  [start] located(ev) == stowage ;
  [all] possesses(ev, item) == TRUE:
  ->FALSE ;
  [end] located(item):= stowage ;
}
```

[Boddy & Bonasso, 2010]

RDDL [Sanner, 2011]

- became the official language of the probabilistic track of the IPC since 2011
- models partial observability
- efficient description of (PO)MDPs

```
domain wildfire_mdp {

types {
x_pos : object;
y_pos : object;
};

pvariables {

// Action costs and penalties
COST_CUTOUT      : {non-fluent, real, default = -5 }; // Cost
to cut-out fuel from a cell
COST_PUTOUT     : {non-fluent, real, default = -10 }; // Cost
to put-out a fire from a cell
PENALTY_TARGET_BURN : {non-fluent, real, default = -100 }; //
Penalty for each target cell that is burning
PENALTY_NONTARGET_BURN : {non-fluent, real, default = -5 }; //
Penalty for each non-target cell that is burning
....
}

cpfs{
burning'?x, ?y) = if ( put-out'?x, ?y) ) // Intervention to
put out fire?
                then false
                // Modification: targets can only start to burn if at
least one neighbor is on fire
                else if (~out-of-fuel'?x, ?y) ^ ~burning'?x, ?y)) //
Ignition of a new fire? Depends on neighbors.
                then [if (TARGET'?x, ?y) ^ ~exists_{?x2: x_pos, ?
y2: y_pos} (NEIGHBOR'?x, ?y, ?x2, ?y2) ^ burning'?x2, ?y2))
                then false
                else Bernoulli( 1.0 / (1.0 + exp[4.5 -
(sum_{?x2: x_pos, ?y2: y_pos} (NEIGHBOR'?x, ?y, ?x2, ?y2) ^
burning'?x2, ?y2)))] ) ]
                else
                burning'?x, ?y); // State persists

...
}
```

https://cs.uwaterloo.ca/~mgrzes/IPPC_2014/



Domain-independent Planners

- Dozens of classical planners
 - support typed STRIPS
 - newer planners support action costs, and some ADL features
 - many of them are optimal
- Several temporal planners
 - support durative actions
 - few support numeric fluents or timed-initial literals
 - few fully support concurrency
 - very few are optimal
- Several probabilistic planners
 - (PO)MDP
 - FOND
- A few continuous planners
-

Language Expressiveness vs. Planning Engines

- *“It is almost a law in PDDL planning that for every language feature one adds to a domain definition, the number of planners that can solve (or even parse) it, and the efficiency of those planners, falls exponentially”* [anonymous reviewer]
- Motivate **development of more expressive** planning engines
- **Reduce** the number of **features** in models



KE Tools for Planning Domain Modelling



Purpose of KE tools

- Assist in domain developing process
 - Support development cycle (as in SW engineering)
 - Visualize (parts of) the model
 - Verification and Validation support (e.g. consistency check)
 - ...
- Usable by non-experts (but with basic knowledge of planning)



GIPO [Simpson et al., 2007]

- GIPO (Graphical Interface for Planning with Objects) won the ICKEPS 2005 competition
- Based on the **OCL** (Object-Centred Language)
- Define **life histories of objects**
- Supports “**classical**” **PDDL** (limitedly also “durative” actions)
- Supports **HTN** (HyHTN planner is integrated) [McCluskey et al., 2003]

ItSimple [Vaquero et al., 2007;2012]

- Supports development cycle
- Exploits **UML** for domain modelling
- Exploits **Petri Nets** for dynamic analysis of **state machines** (e.g. reachability analysis)
- Supports **PDDL 3.1**

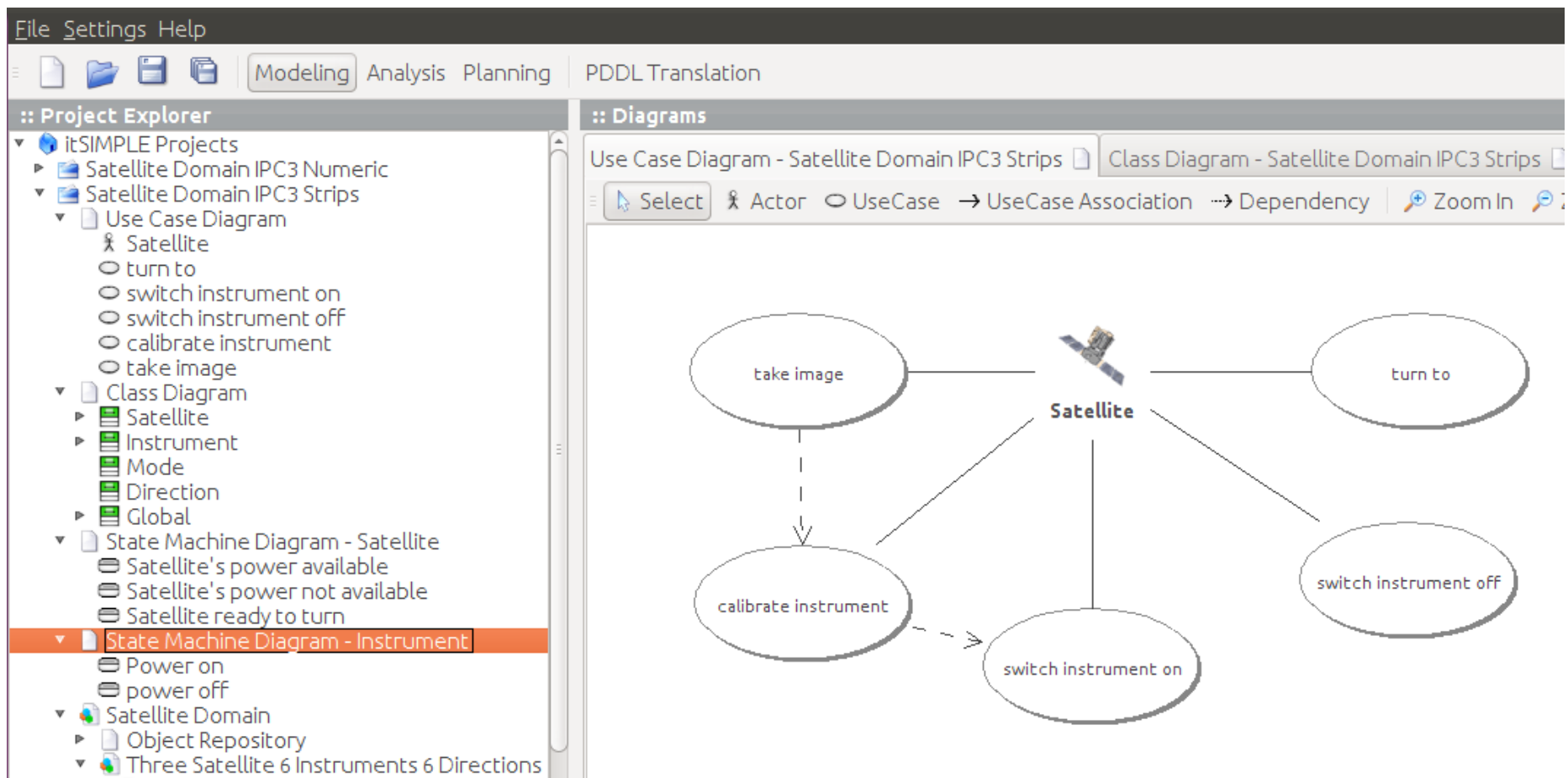
<https://code.google.com/archive/p/itsimple/>

- Project webpage

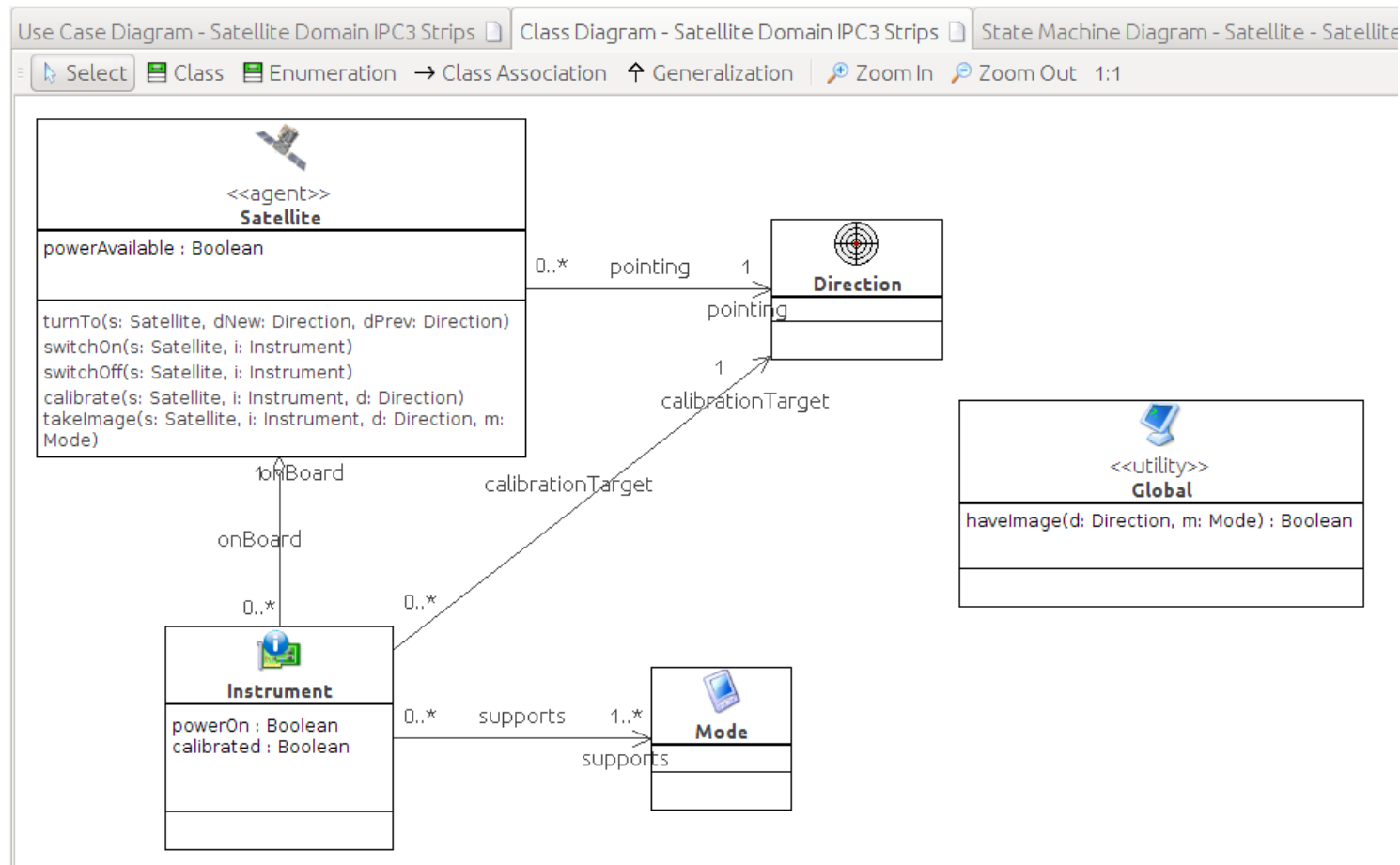
http://www.youtube.com/watch?feature=player_embedded&v=FGBhvBnzyvo

- Tutorial on domain modelling it ItSimple by Chris Muise

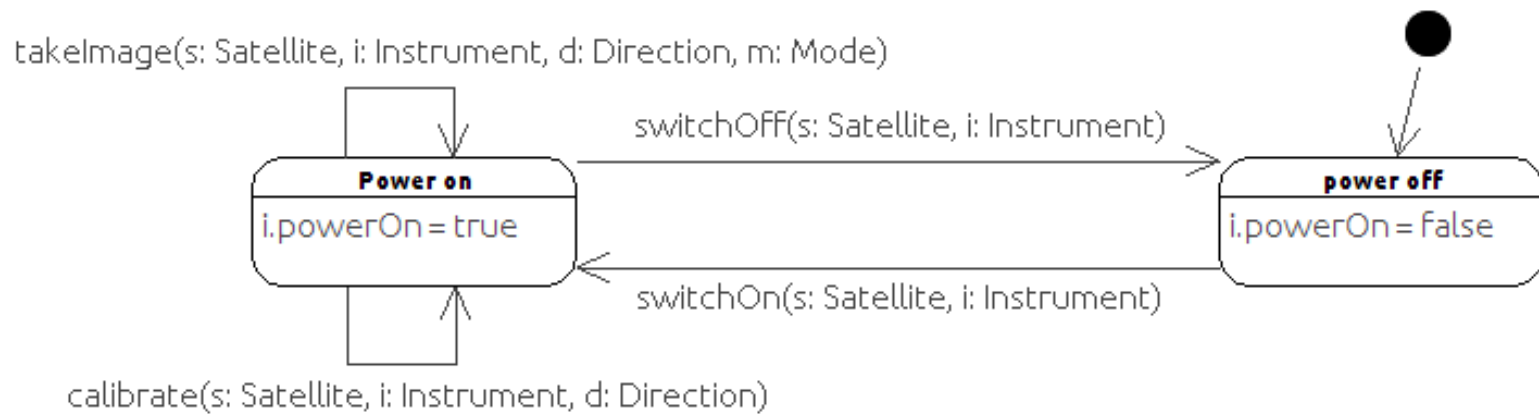
ItSimple - Sample Use Case



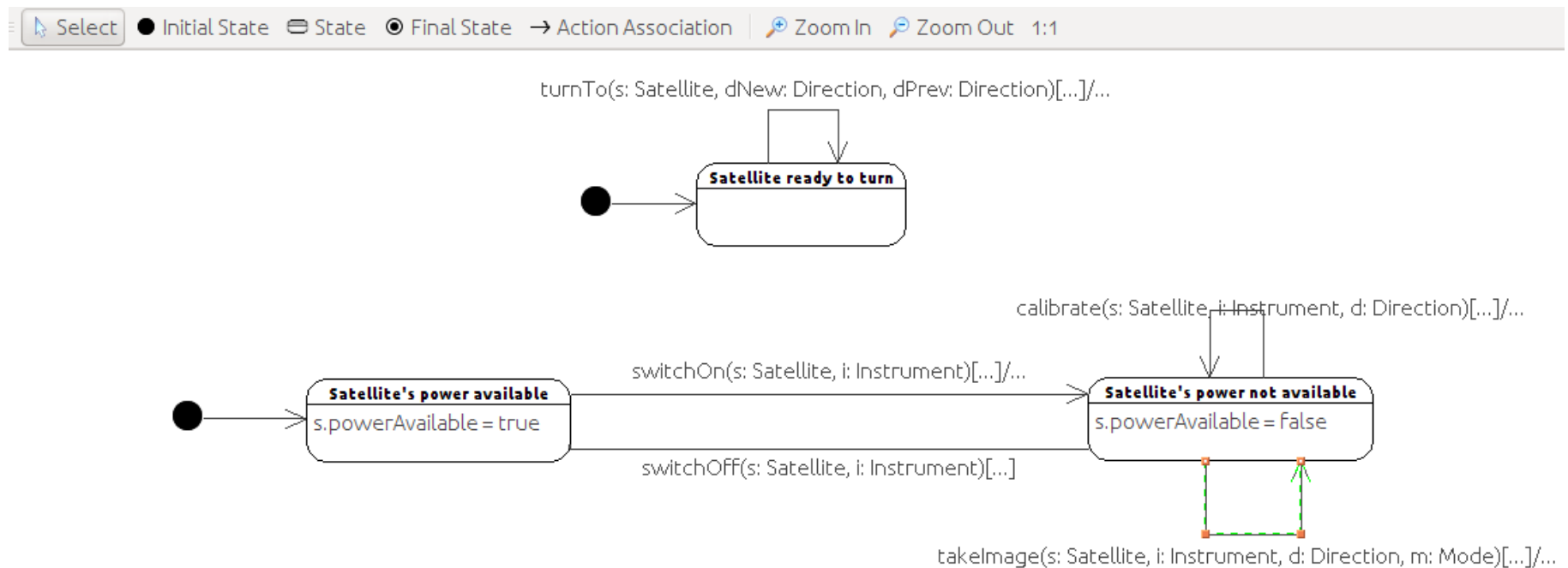
ItSimple - Sample Class Diagram



ItSimple - Sample State Machine (Satellite)



ItSimple - Sample State Machine (Instrument)





Some other KE Frameworks

- EUROPA [Barreiro et al., 2012]
 - Framework supporting NDDL and ANML
- JABBAH [Gonzalez-Ferrer et al., 2009]
 - Supports HTN
- KEWI [Wickler et al., 2014]
 - Object Centred (including inheritance)
 - Web Application (supports collaboration)
- VIZ [Vodrazka & Chrupa, 2010]
 - A “light-weight” KE tool



Planning.Domains

- “A Collection of Tools for Working with Planning Domains” [Muisse]
- Web application
- Rich editor (syntax highlighting, autocomplete, etc.)
- Plug-in support
- Repository of all domains and problems from the IPCs

Planning.Domains - Sample Domain (Satellite)

```
1 (define (domain satellite)
2 (:requirements :equality :strips)
3- (:predicates
4   (on_board ?i ?s) (supports ?i ?m) (pointing ?s ?d) (power_avail ?s) (power_on ?i) (calibrated ?i) (have_image ?d ?m) (calibration_target ?i ?d)(satellite ?x) (direction ?x) (instrument ?x) (mode ?x)
5- (:action turn_to
6   :parameters ( ?s ?d_new ?d_prev)
7   :precondition
8     (and (satellite ?s) (direction ?d_new) (direction ?d_prev) (pointing ?s ?d_prev))
9   :effect
10    (and (pointing ?s ?d_new) (not (pointing ?s ?d_prev))))
11
12- (:action switch_on
13   :parameters ( ?i ?s)
14   :precondition
15     (and (instrument ?i) (satellite ?s) (on_board ?i ?s) (power_avail ?s))
16   :effect
17     (and (power_on ?i) (not (calibrated ?i)) (not (power_avail ?s))))
18
19- (:action switch_off
20   :parameters ( ?i ?s)
21   :precondition
22     (and (instrument ?i) (satellite ?s) (on_board ?i ?s) (power_on ?i))
23   :effect
24     (and (power_avail ?s) (not (power_on ?i))))
25
26- (:action calibrate
27   :parameters ( ?s ?i ?d)
28   :precondition
29     (and (satellite ?s) (instrument ?i) (direction ?d) (on_board ?i ?s) (calibration_target ?i ?d) (pointing ?s ?d) (power_on ?i))
30   :effect
31     (calibrated ?i))
32
33- (:action take_image
34   :parameters ( ?s ?d ?i ?m)
35   :precondition
36     (and (satellite ?s) (direction ?d) (instrument ?i) (mode ?m) (calibrated ?i) (on_board ?i ?s) (supports ?i ?m) (power_on ?i) (pointing ?s ?d) (power_on ?i))
37   :effect
38     (have_image ?d ?m))
39
40 )
41
```

Planning.Domains - Sample Plan (Satellite domain)

The screenshot displays the PDDL Editor interface. The browser address bar shows the URL `editor.planning.domains/#`. The application title is "PDDL Editor". The menu bar includes "File", "Session", "Import", "Solve", "Torchlight", "Plugins", and "Help".

On the left, a file explorer shows a list of files: `unamed1.pddl`, `domain.pddl`, `Analysis (1)`, `poi-pfile1.pddl`, `Analysis (2)`, and `Plan (1)`. The `Plan (1)` file is selected.

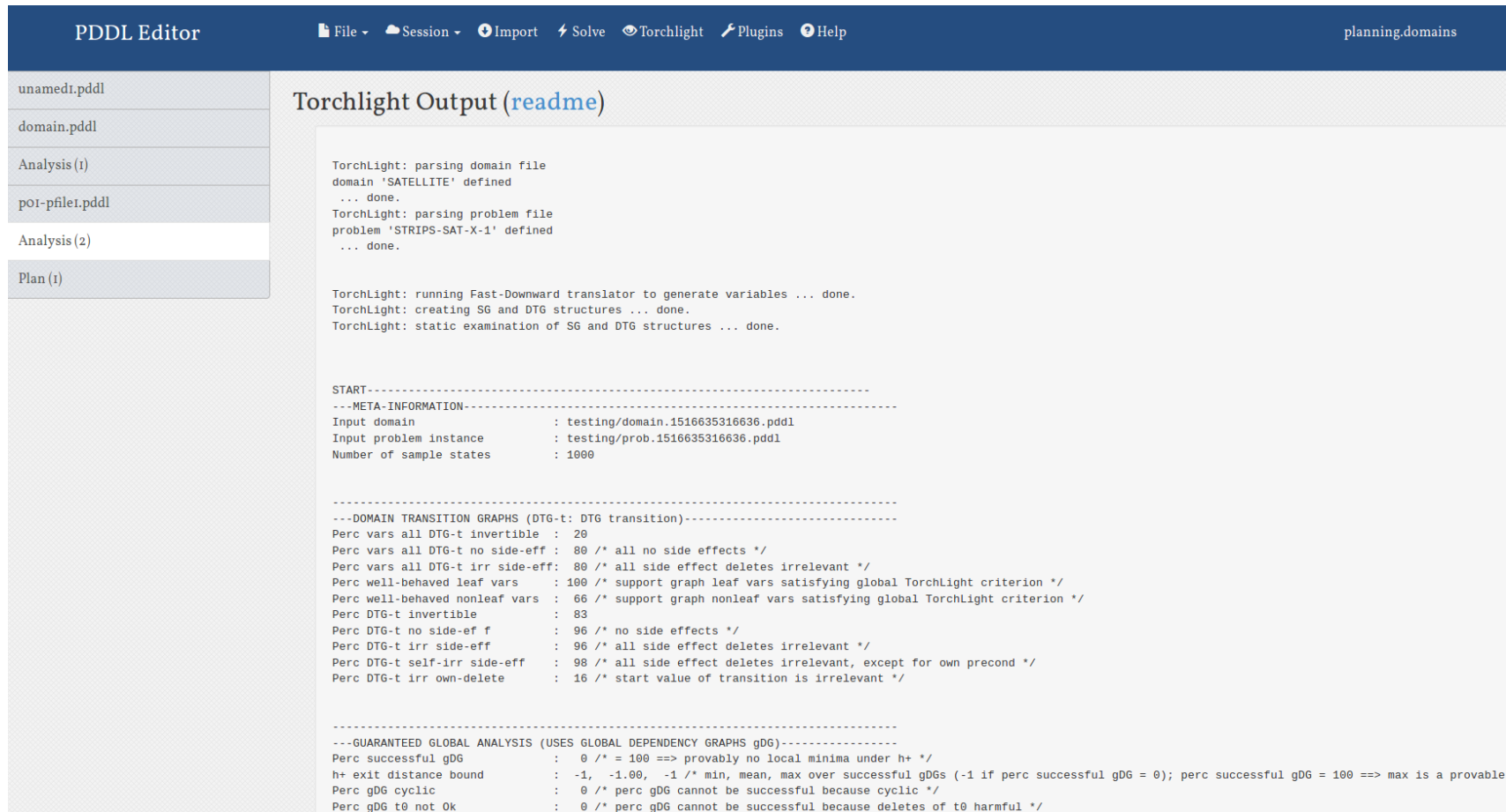
The main area is titled "Found Plan (output)" and contains a list of actions:

- `(turn_to satellite0 groundstation2 phenomenon6)`
- `(switch_on instrument0 satellite0)`
- `(calibrate satellite0 instrument0 groundstation2)`
- `(turn_to satellite0 phenomenon4 groundstation2)` (highlighted in blue)
- `(take_image satellite0 phenomenon4 instrument0 thermographo)`
- `(turn_to satellite0 star5 phenomenon4)`
- `(take_image satellite0 star5 instrument0 thermographo)`
- `(turn_to satellite0 phenomenon6 star5)`
- `(take_image satellite0 phenomenon6 instrument0 thermographo)`

To the right, a code editor shows the following code snippet:

```
(:action turn_to
:parameters (satellite0 phenomenon4 groundstation2)
:precondition
  (and
    (satellite satellite0)
    (direction phenomenon4)
    (direction groundstation2)
    (pointing satellite0 groundstation2)
  )
:effect
  (and
    (pointing satellite0 phenomenon4)
    (not
      (pointing satellite0 groundstation2)
    )
  )
)
```

Planning.Domains - Analysis (by TorchLight)



The screenshot displays the PDDL Editor interface. The top menu bar includes 'File', 'Session', 'Import', 'Solve', 'Torchlight', 'Plugins', and 'Help'. The current session is named 'planning.domains'. On the left, a file explorer shows 'unamed1.pddl', 'domain.pddl', 'Analysis (1)', 'poi-pfile1.pddl', 'Analysis (2)', and 'Plan (1)'. The main area is titled 'Torchlight Output (readme)' and contains the following text:

```
TorchLight: parsing domain file
domain 'SATELLITE' defined
... done.
TorchLight: parsing problem file
problem 'STRIPS-SAT-X-1' defined
... done.

TorchLight: running Fast-Downward translator to generate variables ... done.
TorchLight: creating SG and DTG structures ... done.
TorchLight: static examination of SG and DTG structures ... done.

START-----
---META-INFORMATION-----
Input domain           : testing/domain.1516635316636.pddl
Input problem instance : testing/prob.1516635316636.pddl
Number of sample states : 1000

-----
---DOMAIN TRANSITION GRAPHS (DTG-t: DTG transition)-----
Perc vars all DTG-t invertible : 20
Perc vars all DTG-t no side-eff : 80 /* all no side effects */
Perc vars all DTG-t irr side-eff: 80 /* all side effect deletes irrelevant */
Perc well-behaved leaf vars    : 100 /* support graph leaf vars satisfying global TorchLight criterion */
Perc well-behaved nonleaf vars : 66 /* support graph nonleaf vars satisfying global TorchLight criterion */
Perc DTG-t invertible         : 83
Perc DTG-t no side-ef f      : 96 /* no side effects */
Perc DTG-t irr side-eff     : 96 /* all side effect deletes irrelevant */
Perc DTG-t self-irr side-eff : 98 /* all side effect deletes irrelevant, except for own precondition */
Perc DTG-t irr own-delete   : 16 /* start value of transition is irrelevant */

-----
---GUARANTEED GLOBAL ANALYSIS (USES GLOBAL DEPENDENCY GRAPHS gDG)-----
Perc successful gDG          : 0 /* = 100 ==> provably no local minima under h+ */
h+ exit distance bound      : -1, -1.00, -1 /* min, mean, max over successful gDGs (-1 if perc successful gDG = 0); perc successful gDG = 100 ==> max is a provable
Perc gDG cyclic             : 0 /* perc gDG cannot be successful because cyclic */
Perc gDG t0 not Ok         : 0 /* perc gDG cannot be successful because deletes of t0 harmful */
```

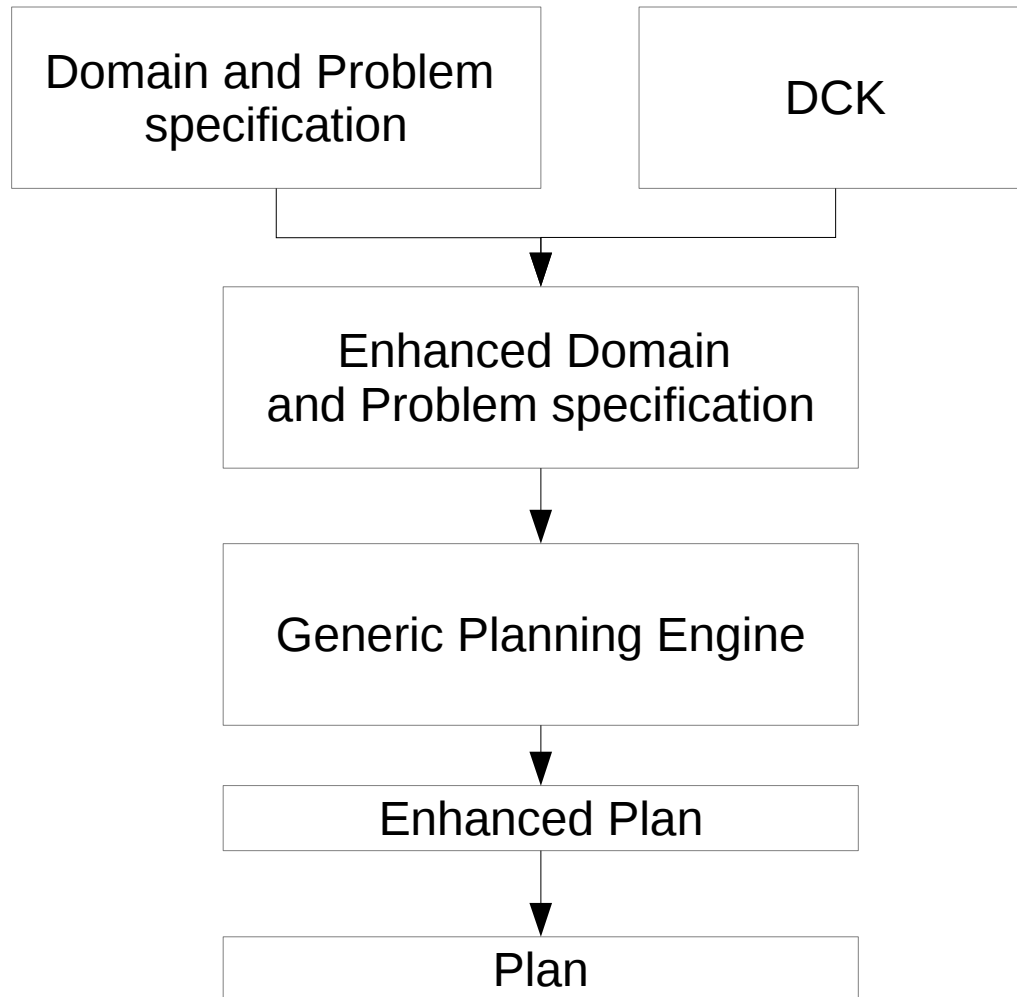


Domain Control Knowledge and Model Reformulation

Domain Control Knowledge (DCK)

- Captures useful domain-specific information
- Provides “guidance” for planning engines
- Complement “raw” domain model specification
- Two main categories of DCK
 - Planner-specific (e.g. TALPlanner, Roller)
 - Planner-independent (this talk !)

Planner-independent DCK





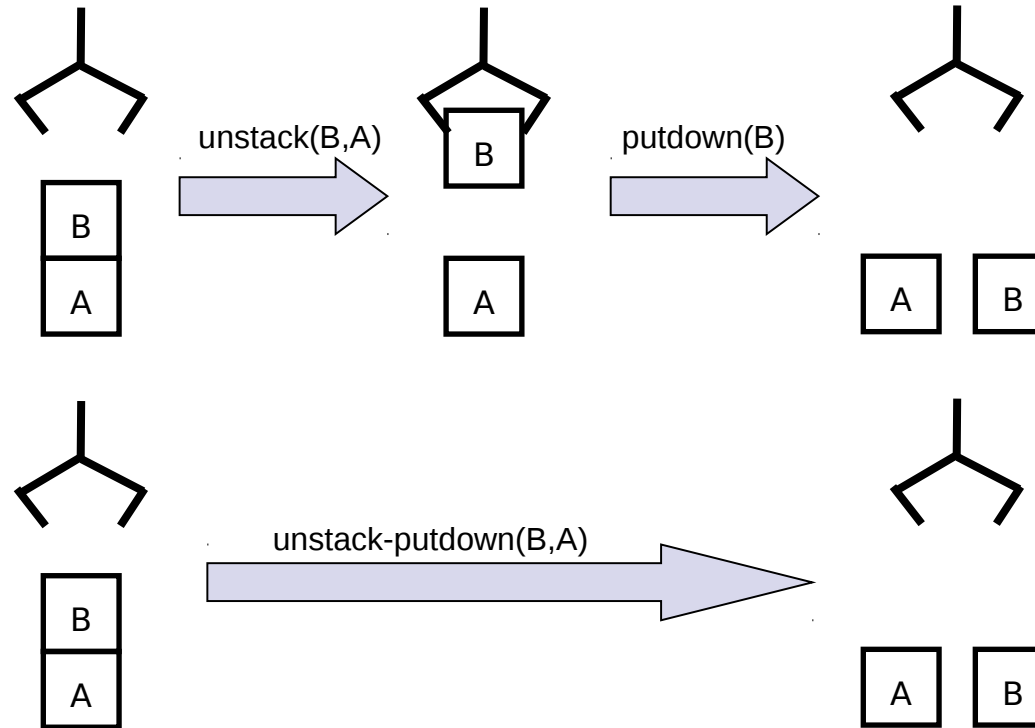
Obtaining DCK

- Automatically
 - training based
 - online
- Manually

Macro-operators (Macros)

- Primitive operators can be assembled into one single operator – macro-operator (macro)
- Assemblage of operators o_i and o_j into $o_{i,j}$:
 - $\text{pre}(o_{i,j}) = \text{pre}(o_i) \cup (\text{pre}(o_j) - \text{add}(o_i))$
 - $\text{del}(o_{i,j}) = (\text{del}(o_i) - \text{add}(o_j)) \cup \text{del}(o_j)$
 - $\text{add}(o_{i,j}) = (\text{add}(o_i) - \text{del}(o_j)) \cup \text{add}(o_j)$
- Widely studied (e.g. Macro-FF, Wizard, MUM, BLOMA)

Macros - example



```
unstack(X,Y) =
{ {on(X,Y),clear(X),handempty} //prec
  {on(X,Y),clear(X),handempty} //neg eff
  {holding(X),clear(Y)} } //pos eff
```

```
putdown(X) =
{ {holding(X)} //prec
  {holding(X)} //neg eff
  {ontable(X),on(X),clear(X),handempty} } //pos eff
```

```
unstack-putdown(X,Y) =
{ {on(X,Y),clear(X),handempty} //prec
  {on(X,Y),holding(X),} //neg eff
  {clear(X),clear(Y),ontable(X),handempty} } //pos eff
```

Macros - Benefits and Shortcomings

- Macros can be understood as '**short-cuts**' in the search space
- Solution plans can be **much shorter**
- Introducing macros can **increase branching factor** considerably !
- There might be **high memory requirements** for planners



“A short-cut is the longest way between two points”



Outer Entanglements [Chrupa & McCluskey 2012]

- Outer entanglements are relations between planning operators and initial or goal predicates
- **Entanglement by init** – allows only such instances of an operator requiring an initial predicate
- **Entanglement by goal** – allows only such instances of an operator achieving goal predicates

```

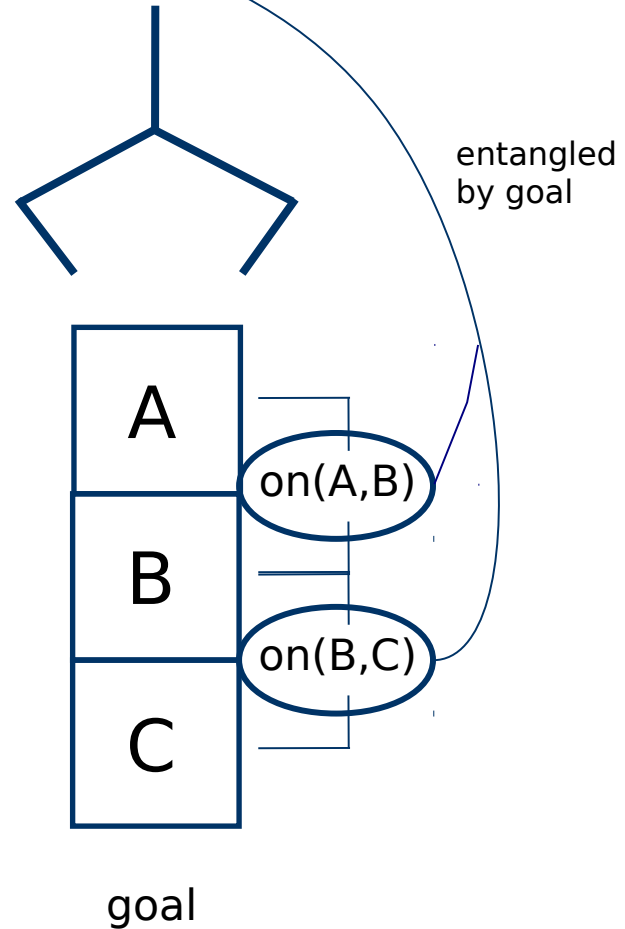
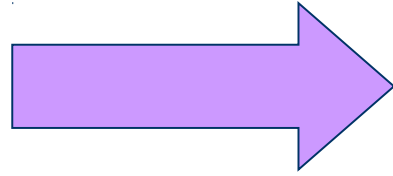
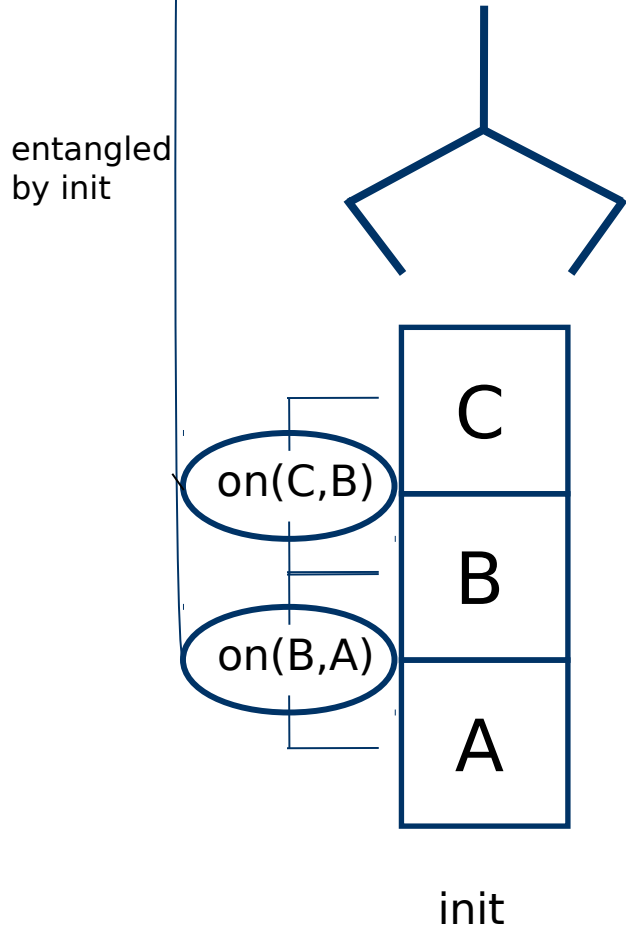
Unstack(X,Y) =
( on(X,Y), clear(X), handempty } //prec
{ on(X,Y), clear(X), handempty } //neg eff
{ holding(X), clear(Y) } //pos eff

```

```

Stack(X,Y) =
( { holding(X), clear(Y) } //prec
{ holding(X), clear(Y) } //neg eff
on(X,Y), clear(X), handempty } //pos eff

```



allowed: Unstack(C,B), Unstack(B,A)

allowed: Stack(A,B), Stack(B,C)



Outer Entanglements - benefits and shortcomings

- Outer Entanglements **restrict** the number of instantiated operators
- Outer Entanglements (significantly) **reduces** memory requirements
- The method for extracting outer entanglements **does not ensure completeness**

Combining Macros and Outer Entanglements

- MUM [Chrupa et al., 2014]
 - Outer entanglements can **reduce branching factor** the macros introduce
 - Applying outer entanglements only on macros **does not compromise completeness**
 - Outer entanglements **provide heuristics** in the macro learning process
- OMA [Chrupa et al., 2015] – an online version of MUM

Transition-based DCK [Chrupa & Bartak, 2016]

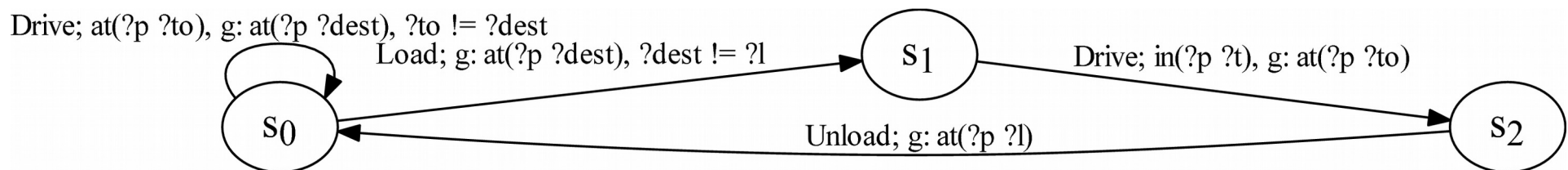
- Inspired by Finite State Automata
- Define “grammar” of solution plans
- “Schematical” representation is easier to understand by non-experts in planning
- Can be incorporated in planning domain models

Transition-based DCK - formal specification

- A quadruple (S, O, T, s_0) where
 - S is a set of **DCK states**
 - $s_0 \in S$ is the **initial DCK state**
 - O is a set of **planning operators**
 - T is a set of **transitions**
- Each transition is in the form (s, o, C, s') where
 - $s, s' \in S, o \in O$
 - C is a set of **constraints** where each is in the form
 - $p, \neg p$ – p **must or must not be** in the current planning state
 - $g: p$ – p must be **an open goal** in the current planning state

Specifying Transition-based DCK - an example

- An empty truck (can carry at most one package) should move only to locations where some package is waiting to be delivered
- After a package that has to be delivered is loaded into the truck, the truck moves to package's goal location where the package is then unloaded





Impact of DCK

- Macros and Entanglements **have considerable impact on performance** in some cases
- Transition-based DCK in some cases “**determinize**” the planning process
- Changes in the domain model might require **considerable changes** in DCK

Impact of DCK on the KE process

- In practice, separating the “raw” domain model and DCK is easier to maintain
- Extend existing KE tools (e.g. itSimple, Planning.Domains) by supporting automatic/manual DCK acquisition
- Understanding in which cases planners fail and how DCK can alleviate such an issue
 - Even changing the order of operators and predicates in their preconditions/effects have a significant impact on planners' performance !



Conclusions

- KE process in planning is still “black art”
 - No guidelines/methodologies
 - Little support of KE tools
 - Effective DCK acquisition support
- A little to nothing has been done in non-classical planning
- **Addressing these issues will significantly strengthen the position of domain-independent planning in other AI areas**