Knowledge Engineering in Planning

Representation Matters

Lukáš Chrpa Mauro Vallati

Czech Technical University in Prague & Charles University in Prague

University of Huddersfield

Table of contents

Introduction

Introduction: What's Planning?

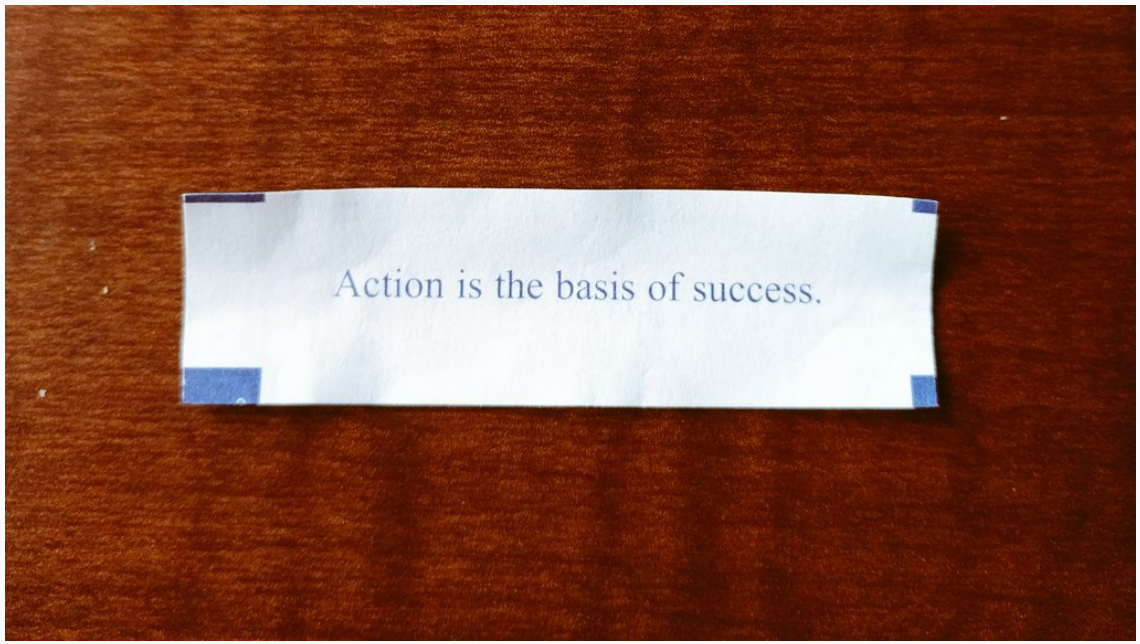
Planning is a pivotal element of what we call **intelligence**

The scope of **AI Planning** is the synthesis (generation) and execution of PLANS.

AI Planners reason with actions, and information about the world and the goals, and generate plans.

Introduction: Importance of Planning

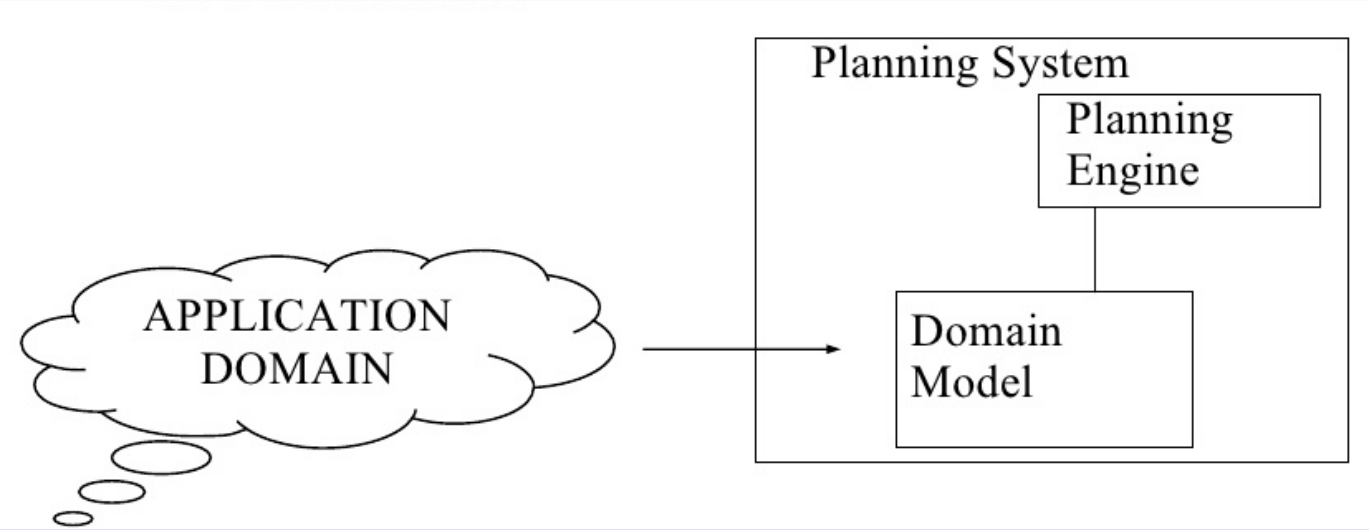
I don’t think we really need any further discussion on the importance of planning.



Introduction: Main Assumption

The MAIN ASSUMPTION of what is termed *domain-independent* AI Planning is that there should be a logical separation between

- The Planning Engine
- The Domain Model



Introduction: Main Assumption

(**Positive**) Implications of assumption:

- It tends to make AI Planning a different subject to eg “planning and scheduling” in manufacturing
- planning engines AND domain models can be developed, tested, debugged, validated independently – the *model* of the particular application of planning is developed in relative isolation to the planning engine
- domain models may be useful for more purposes than simply automated planning functions (e.g., validation, mining, etc.)

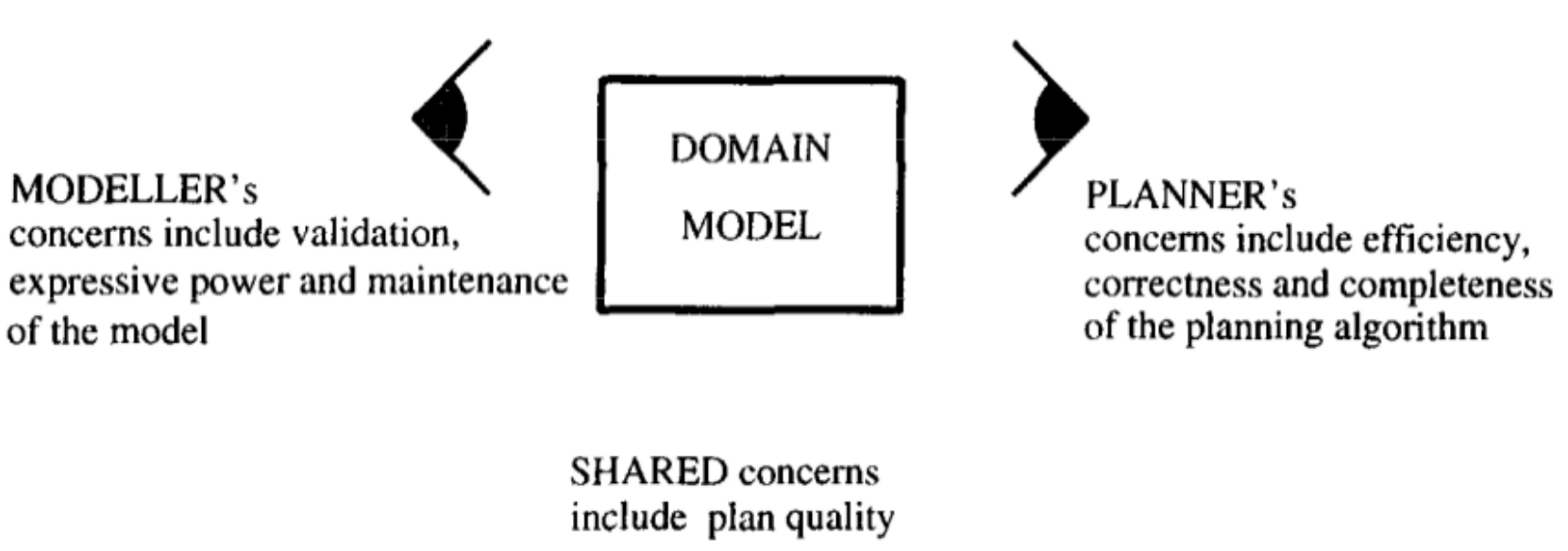
Introduction: Main Assumption

(**Negative**)Consequences of assumption:

- Domain model representation is influenced by ‘what planners can handle’
- Inefficiency in domain-independent planning systems on a specific domain of interest

Much research work has been carried out in developing planning engines, pushed along by the International Planning Competitions, **BUT** methods and tools to help develop the domain models have had relatively little attention

Introduction: Involved Perspectives



(taken from [27])

Extraction of Domain Knowledge, under the form of reformulation, can help supporting both sides.

Language(s) and Planners

The Planning Domain Description Language (PDDL)

PDDL has been introduced for the IPC 1998 and is now widely used in the planning community

- PDDL 1.2 [30] derives from STRIPS, i.e., it offers “predicate centric” representation (or classical representation), and supports ADL (e.g. conditional effects, quantifiers), domain axioms and hierarchical actions
- PDDL 2.1 [15] introduced numeric fluents and durative actions
- PDDL 3.1 [18] introduced trajectory constraints, preferences and object fluents (aka state variable representation)
- PDDL+ [16] introduced continuous processes and events
- PPDDL [45] introduced probabilistic action effects
- MA-PDDL [26] introduced planning by and for multiple agents

Other languages

NDDL [17]

- variable representation
- timelines/activities
- rich temporal constraints

ANML [38]

- notions of states and actions (as in PDDL)
- variable representation (as in NDDL)
- rich temporal constraints (an in NDDL)

Other languages (cont.)

RDDL [34]

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

PICAT [46]

- a “Prolog-like” rule-based language
- structured state representation
- supports control knowledge

PDDL Features Support

Is there any planner supporting all PDDL features ?

PDDL Features Support

Is there any planner supporting all PDDL features ?

NO

PDDL Features Support

Is there any planner supporting all PDDL features ?

NO

Is there any planner supporting all PDDL 1.2 features ? (incl. axioms and hierarchical actions)

PDDL Features Support

Is there any planner supporting all PDDL features ?

NO

Is there any planner supporting all PDDL 1.2 features ? (incl. axioms and hierarchical actions)

NO

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
  - few fully support concurrency
  - very few are optimal
- Several probabilistic planners
- A few continuous planners
- ....

Consequences for Applications of Domain-independent Planning

- “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)

- Hence, it is advisable to **minimize the number of required features** for the model (ideally having just typed STRIPS)
- and **development of more expressive planning engines** should be encouraged

Knowledge Engineering in Planning

KE in Planning: an Overview

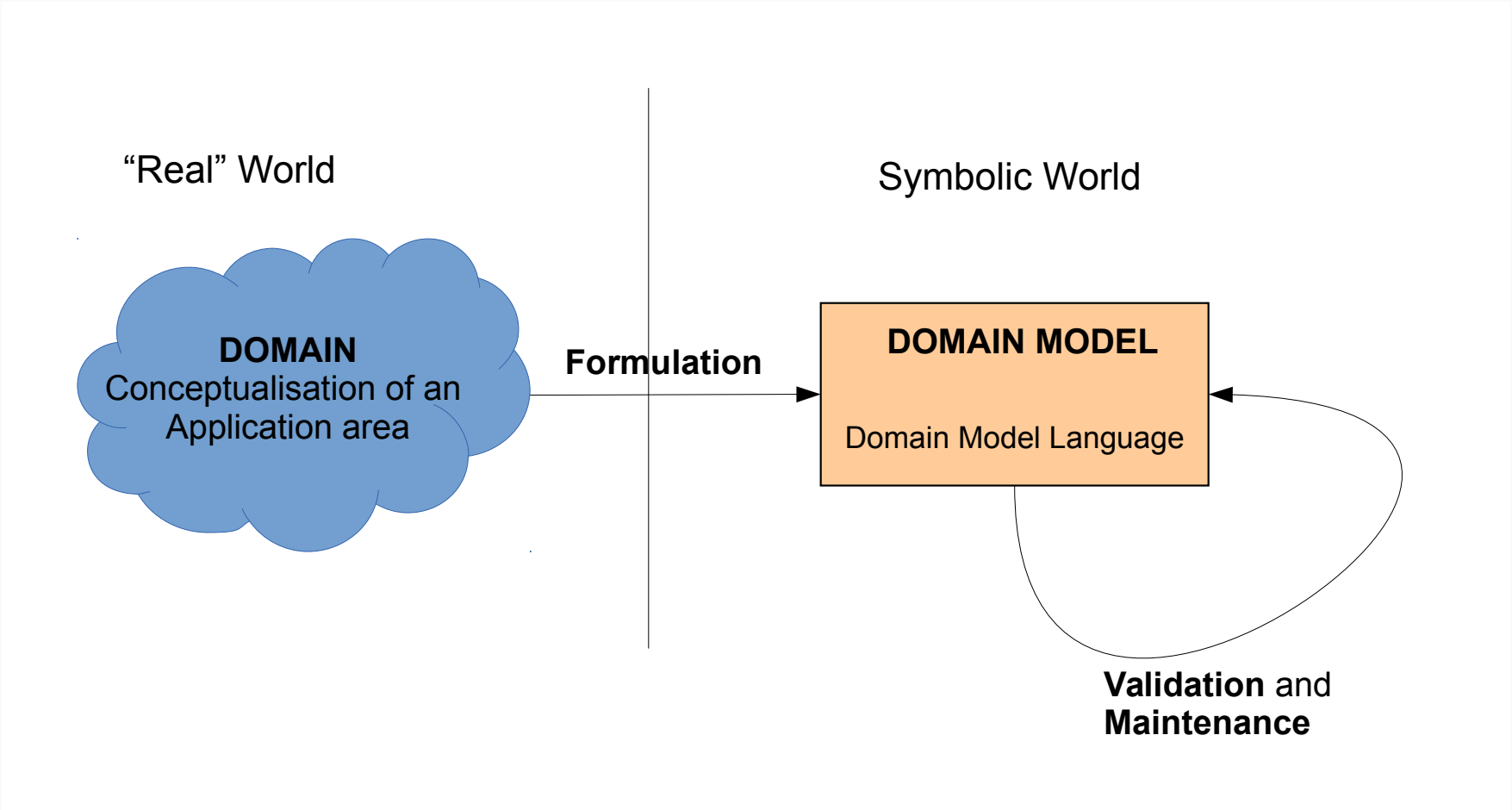
Knowledge Engineering for automated planning is the process that deals with:

- acquisition
- formulation
- validation
- maintenance

of planning domain models.

KE deals also with the selection and optimization of appropriate planning machinery to work on it.

KE in Planning: an Overview



KE in Planning: the Domain Model

> *A planning domain model is a* ***formal specification*** *of the application domain part of the requirements specification which represents entities invariant over every planning problem, such as object classes, functions, properties, relations, and actions in the domain.*

(Taken from [29]; in line with terminology from general KE, works on domain "theories" [7] and the "domain file" in PDDL [30]).

Properties of Domain Models

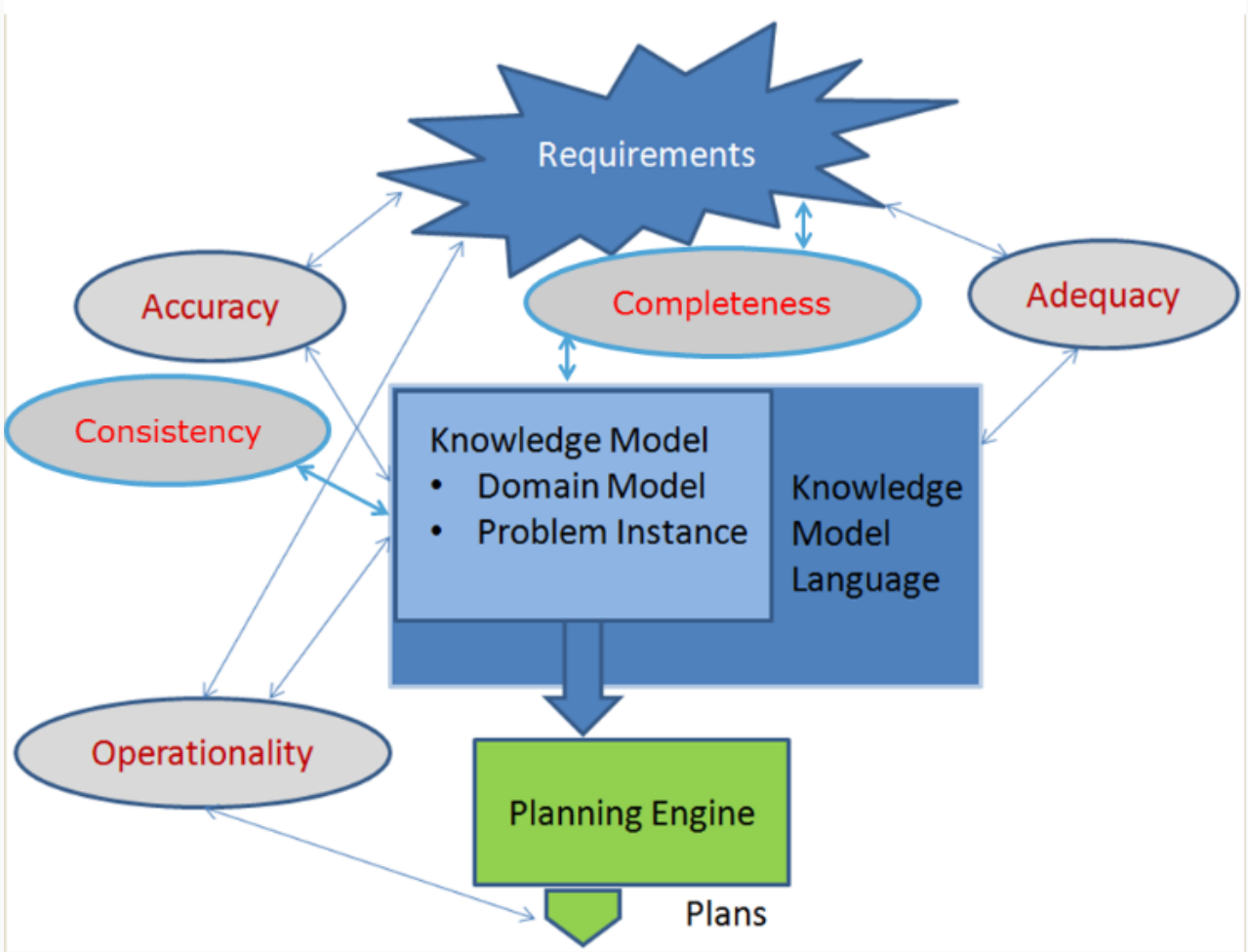


Figure taken from [29].

Properties of Domain Models (2)

- **Accuracy**: there exists a mapping between requirements and model's components.

Properties of Domain Models (2)

- **Accuracy**: there exists a mapping between requirements and model's components.
- **Consistency**: a domain model is consistent if some interpretation exists that makes all its assertions true. (Subcase of the accuracy)

Properties of Domain Models (2)

- **Accuracy**: there exists a mapping between requirements and model's components.
- **Consistency**: a domain model is consistent if some interpretation exists that makes all its assertions true. (Subcase of the accuracy)
- **Completeness**: (*very* informally) any solution in the domain model is also a solution for the real-world domain, and vice-versa.

Properties of Domain Models (2)

- **Accuracy**: there exists a mapping between requirements and model's components.
- **Consistency**: a domain model is consistent if some interpretation exists that makes all its assertions true. (Subcase of the accuracy)
- **Completeness**: (*very* informally) any solution in the domain model is also a solution for the real-world domain, and vice-versa.
- **Adequacy**: the exploited language has the expressive power to represent the requirements (a model can be accurate, but not adequate).

Properties of Domain Models (2)

- **Accuracy**: there exists a mapping between requirements and model's components.
- **Consistency**: a domain model is consistent if some interpretation exists that makes all its assertions true. (Subcase of the accuracy)
- **Completeness**: (*very* informally) any solution in the domain model is also a solution for the real-world domain, and vice-versa.
- **Adequacy**: the exploited language has the expressive power to represent the requirements (a model can be accurate, but not adequate).
- **Operationality**: there is a planning engine that can produce solutions within acceptable resource bounds.

KE in Planning: Domain Model Development

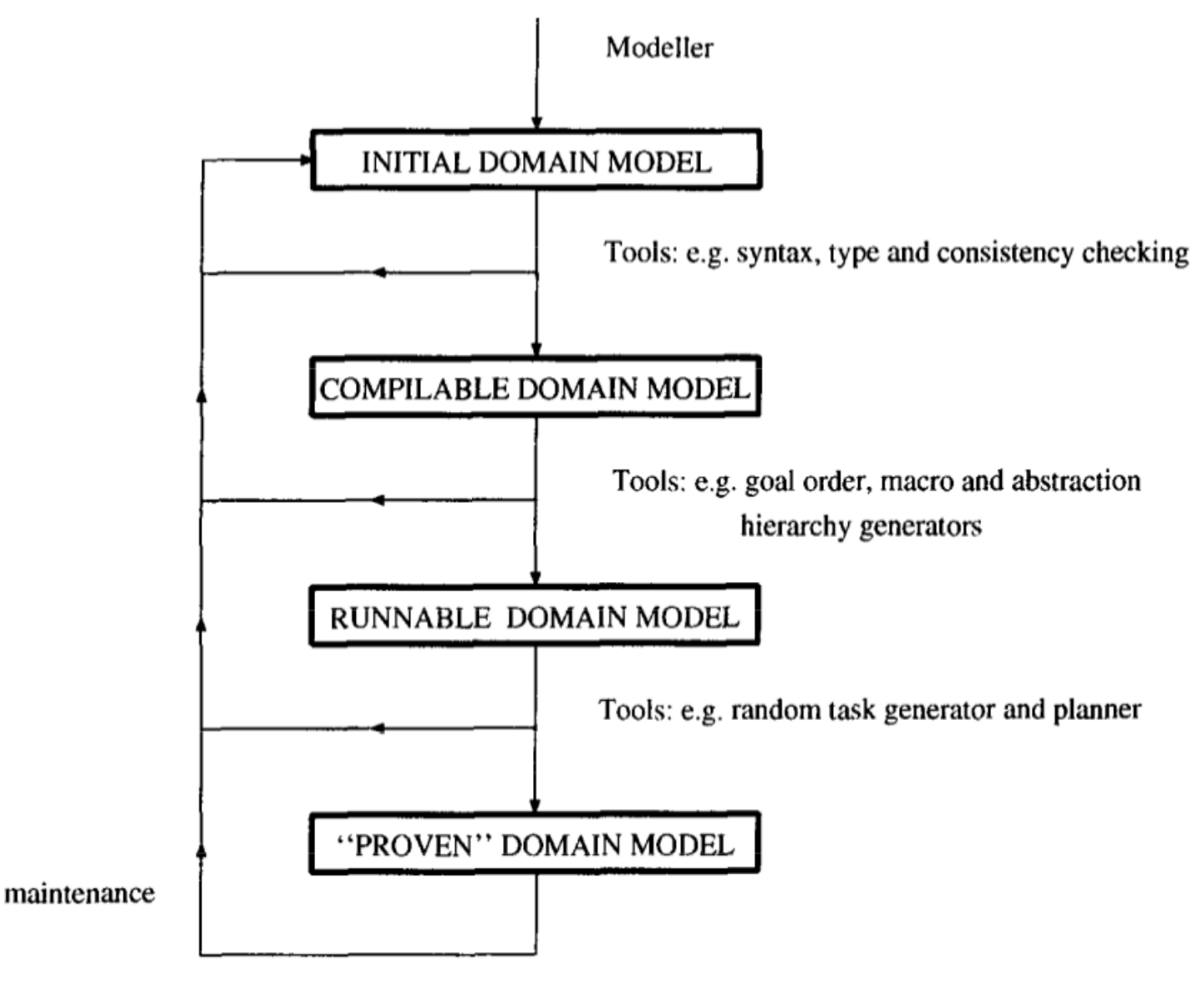


Figure taken from [27].

Tools for KE in Planning

A short and by no mean comprehensive list of existing tools for KE.

- Frameworks
  - **EUROPA** [5]: Integrated platform, able to handle ANML and NDDL.
  - **GIPO** [37]: based on Object-centred language, allowing consistency check.
  - **itSIMPLE** [41]: Exploits UML for object-oriented modeling.
  - **JABBAH** [19]: based on HTN.
  - **MARIO** [14]: tag-based representation language for composition of problems and goals.
  - **VIZ** [42]: exploits diagrams to support PDDL model encoding.
  - **KEWI** [43]: Ontology-based, for non-planning experts.
- PDDL editors
  - **PDDL Studio** [33]
  - **planning.domains**

Usefulness of Tools

Recently [35], a few KE approaches for planning were compared.

- Method A: a PDDL expert that uses a text editor
- Method B: a user that exploits itSIMPLE
- Method C: a PDDL experts using GIPO

Three real-world domains considered:

- Road accident management
- Machine tool calibration
- Urban traffic control

Usefulness of Tools: Criteria

Operationality. How efficient are the models produced?

Collaboration. Does the tool help in team efforts?

Maintenance. How easy is it to come back and change a model?

Experience. Is the tool indicated for inexperienced planning users?

Efficiency. How quickly are acceptable models produced?

Debugging. Does the tool support debugging?

Support. Are there manuals available for using the tool?

Usefulness of Tools: Identified Issues

- **Expertise**: current KE approaches require a specific expertise (in planning, UML, OCL, ...)
- **Team work**: no support for team working.
- **Maintenance**: tools do not support the writing of documentation. Some tools are not able to handle models that have been modified.
- **Debugging**: lack of effective techniques for static/dynamic debugging.
- **Language support**: most tools have limited support of PDDL features (and versions).

Promoting KE in P&S: ICKEPS

International **C**ompetition on **K**nowledge **E**ngineering for **P**lanning and **S**cheduling

> *promote the knowledge-based and domain modelling aspects of AI P&S, to accelerate knowledge engineering research, to encourage the development and sharing of prototype tools or software platforms that promise more rapid, accessible, and effective ways to construct reliable and efficient P&S systems*

ICKEPS: History

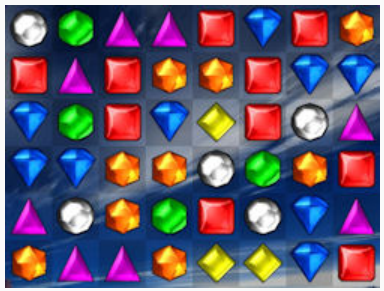
- **ICKEPS 2005** (San Francisco) - Tools and Tools Environments for KE
- **ICKEPS 2007** (Providence) - teams working on KE tasks and application scenarios given out before ICAPS
- **ICKEPS 2009** (Thessaloniki) - Tools for translating into planner-ready language from application-oriented language
- **ICKEPS 2012** (Sao Paulo) - teams working on KE tasks and application scenarios given out before ICAPS
- **ICKEPS 2016** (London) teams working on KE tasks and application scenarios given onsite with a fixed period of time

ICKEPS 2016: Structure

A team competition (max size 4) composed by the following stages:

- Pre-competition
- On-site modelling
- Demonstration
- Judges evaluation and assessment



ICKEPS 2016 Evaluation: Metrics

- KE process
  - Tools
    - Innovation and Originality of Tools
    - Usefulness and Support
  - Team Working
    - Strategy
    - Collaboration
- Models (each scenario assessed singularly)
  - Correctness
  - Readability
  - Generality
  - Originality

Models have been evaluated “dynamically”: by extracting basic information (number of operators, ...) and by running a number of planners on the generated problems.

ICKEPS 2016: The Take-home Message

- Large variation in the PDDL models produced, resulting in significant variations of planners' performance, even on toy-sized instances. (+)
- Low number of entries compared to IPCs. (-)
- Most teams did not use any KE tools. (-)
- Existing tools do not effectively support cooperation. (-)

We are still missing an agreed and usable notion of "**Quality**" of models!

Domain Knowledge for Planners

Domain Knowledge - Definition

"Domain Knowledge (DK) is valid knowledge used to refer to an area of human endeavour, an autonomous computer activity, or other specialized discipline."

In planning, domain knowledge captures domain-specific information providing a "guidance" for planning engines.

Categories of Domain Knowledge

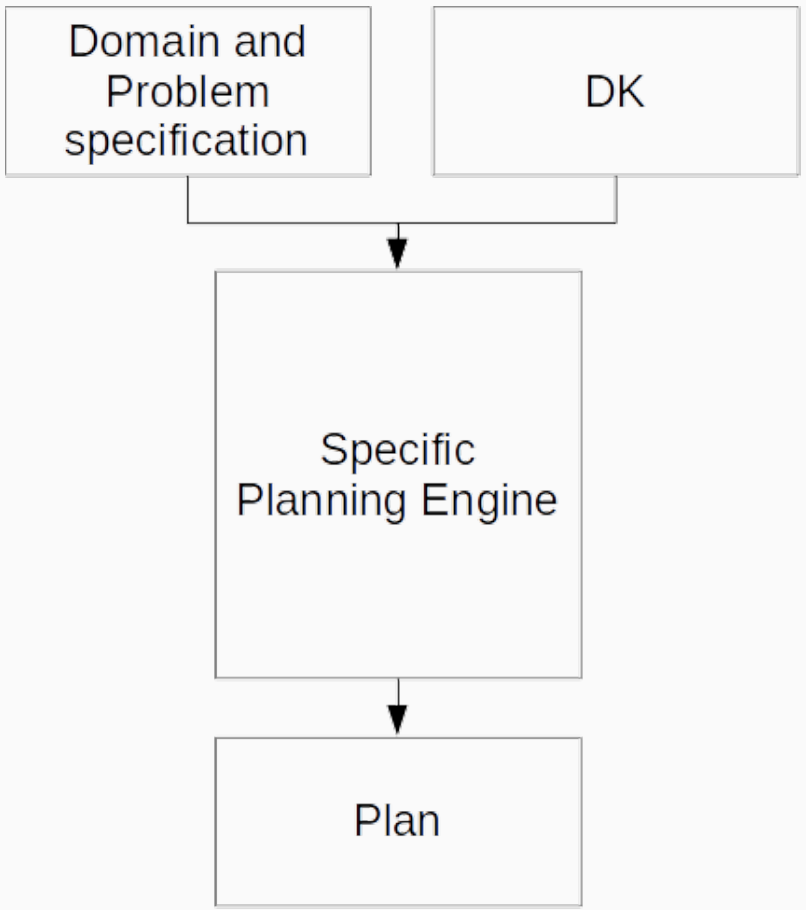
By use

- Planner-specific – a specialized planner has to be used to exploit DK (e.g., TALPlanner, Roller)
- Planner-independent – any generic planner can exploit DK

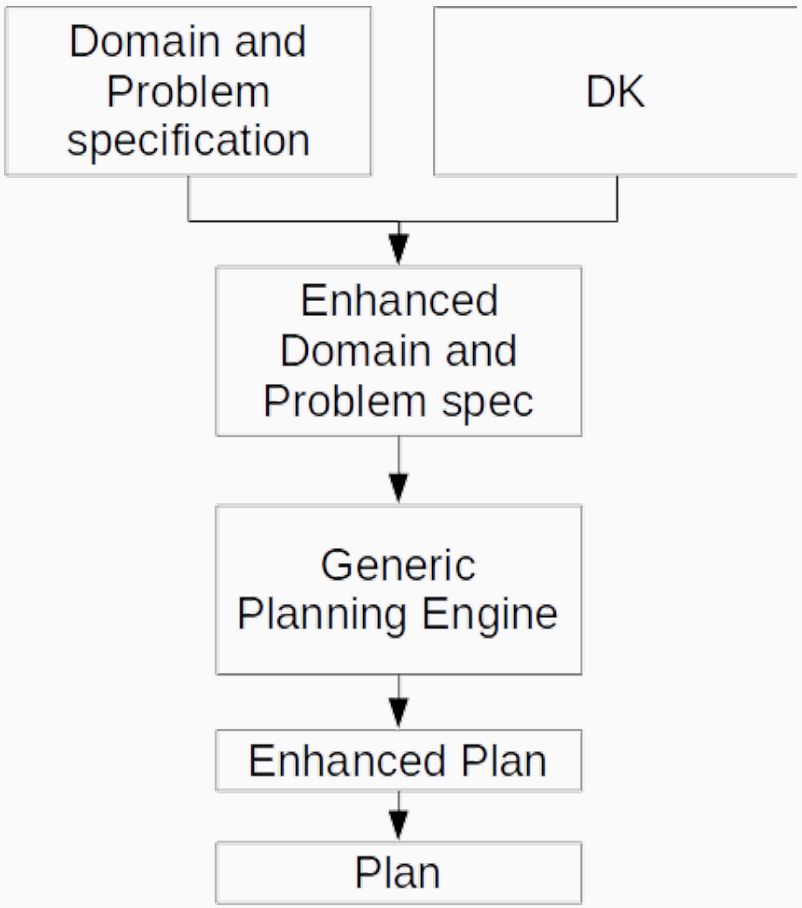
By specification

- Manually specified
- Automatically acquired
- Combination of both

Planner-specific vs. Planner-independent DK



Planner Specific



Planner Independent

Why should we distinguish between Domain Model and Domain Knowledge (in a planner-independent scenario) ?

The RPG Domain

- One of the ICKEPS 2016 scenarios
- A hero has to navigate through a dungeon, where:
  - Rooms are connected by corridors
  - Each room can be empty, with a monster, with a trap, or with a sword
  - The hero can visit a room at most once (after hero's visit the room is destroyed)
- The hero can perform the following actions:
  - **Move** to an adjacent room (not yet visited)
  - **Pick-up** a sword (if it is in the current room)
  - **Destroy** the sword (if the hero carries it)
  - **Disarm** trap (if the trap is in the current room and the hero is empty handed)
- The hero dies if:
  - the hero is empty handed in a room with a monster
  - the hero performs any other action than "disarm" in a room with a trap

RPG domain - the Move action

The hero dies as a result of the Move action if:

- s/he is empty handed and arrives in a room with a monster
- s/he tries to move away from a room with an active trap

The competitors modelled the Move action as:

- MoveWithSword – the hero carries a sword and can enter rooms without traps
- MoveWithoutSword – the hero is empty handed and can enter rooms without monsters

RPG Domain – Observations

The competitors' models "avoid" hero's death

- The models encode DK (i.e., avoiding hero's death is good)
- Planners thus avoid dead-end states
- The models, however, do not fully correspond with the specification

How easy/difficult is to modify such models if something changes in the specification ?

RPG Domain – Lessons Learnt

Planning experts tend to encode domain models in a “planner friendly” way

Such models, however, do not (fully) correspond with the requirements and might be hard to understand for planning non-experts

RPG Domain – Lessons Learnt

Planning experts tend to encode domain models in a “planner friendly” way

Such models, however, do not (fully) correspond with the requirements and might be hard to understand for planning non-experts

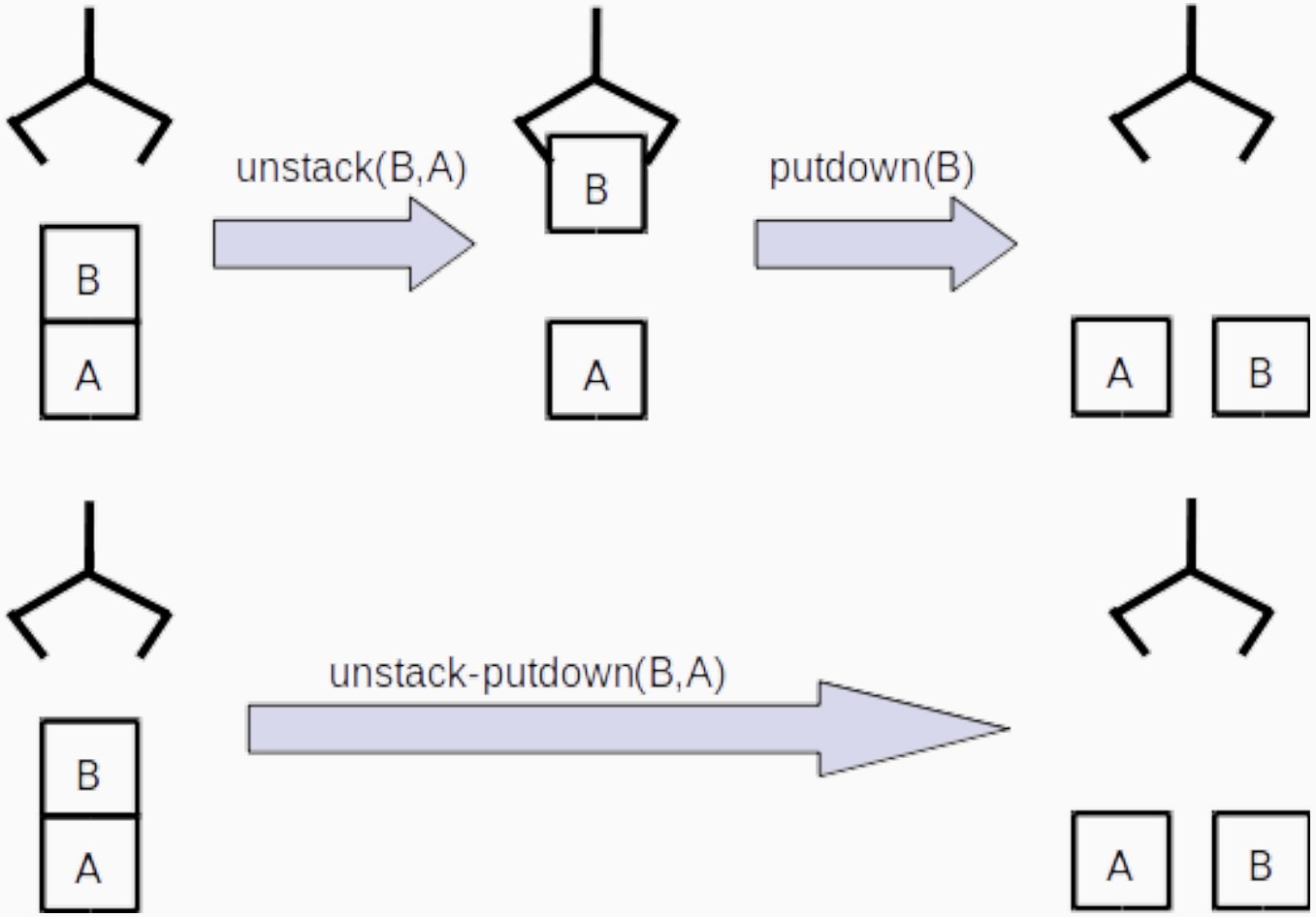
Separating “raw” domain models and DK would be beneficial for both humans and planning engines

Types of Planner-independent Domain Knowledge

Macro-operators

- Ordinary operators can be assembled into **macro-operators (macros)**
- Application of an instance of a macro has the same result as the consecutive application of corresponding instances of ordinary operators
- Macros represent frequent sequences of ordinary operators in plans
- Macros have the same structure as ordinary operators, hence they can be easily added into domain models
- Widely studied (e.g., Macro Problem Solver [25], Macro-FF [6], Wizard [32], MUM [12])

Macros - example



Macros - example (PDDL)

```
(:action unstack
   :parameters (?x - block ?y - block)
   :precondition (and (on ?x ?y)(clear ?x)(handempty))
   :effect (and (holding ?x)(clear ?y)(not (clear ?x))
                (not (handempty))(not (on ?x ?y)))
)

(:action put-down
   :parameters (?x - block)
   :precondition (and (holding ?x))
   :effect (and (not (holding ?x))(clear ?x)(handempty)(ontable ?x))
)

(:action unstack_put-down
   :parameters (?x - block ?y - block)
   :precondition (and (clear ?y)(on ?y ?x)(handempty))
   :effect (and (clear ?x)(clear ?y)(ontable ?y)(handempty)
                (not (on ?y ?x))(not (holding ?y)))
)
```

Macros: Planner-independent Approaches

MacroFF [6]

- Defines five rules for generating macros (incl. Abstract Components)
- Select the top k frequent macros in training plans

Wizard [32]

- Exploits genetic programming to generate macros
- Macros are learnt specifically for a given planner

Investigating action dependencies and independencies [8]

- Considers also action subsequences interleaved by independent actions
- Removes original operators that are replaced by macros in training plans

Problem decomposition [3]

- Achieves each goal atom separately and considers “single goal” plans as macros

Macros: Benefits and Drawbacks

+ “Short-cuts” in a search space – searching for a plan might not go that deep
+ Can easily generalize (if parametrized) for larger problem instances
+ Macros do not affect soundness and completeness of the model

- Branching factor can considerably increase
- High memory requirements for planning engines

“Anti” Macro Approaches

Removing “redundant” actions [21]

- Removing actions whose effects can be achieved by applying sequences of other actions
- Aims at simplifying state space (by removing transitions)
- Reduces branching factor

Splitting Action Schema [2]

- Decomposing more complex actions into simpler ones
- Aims at reducing the grounded model size
- Reduces branching factor

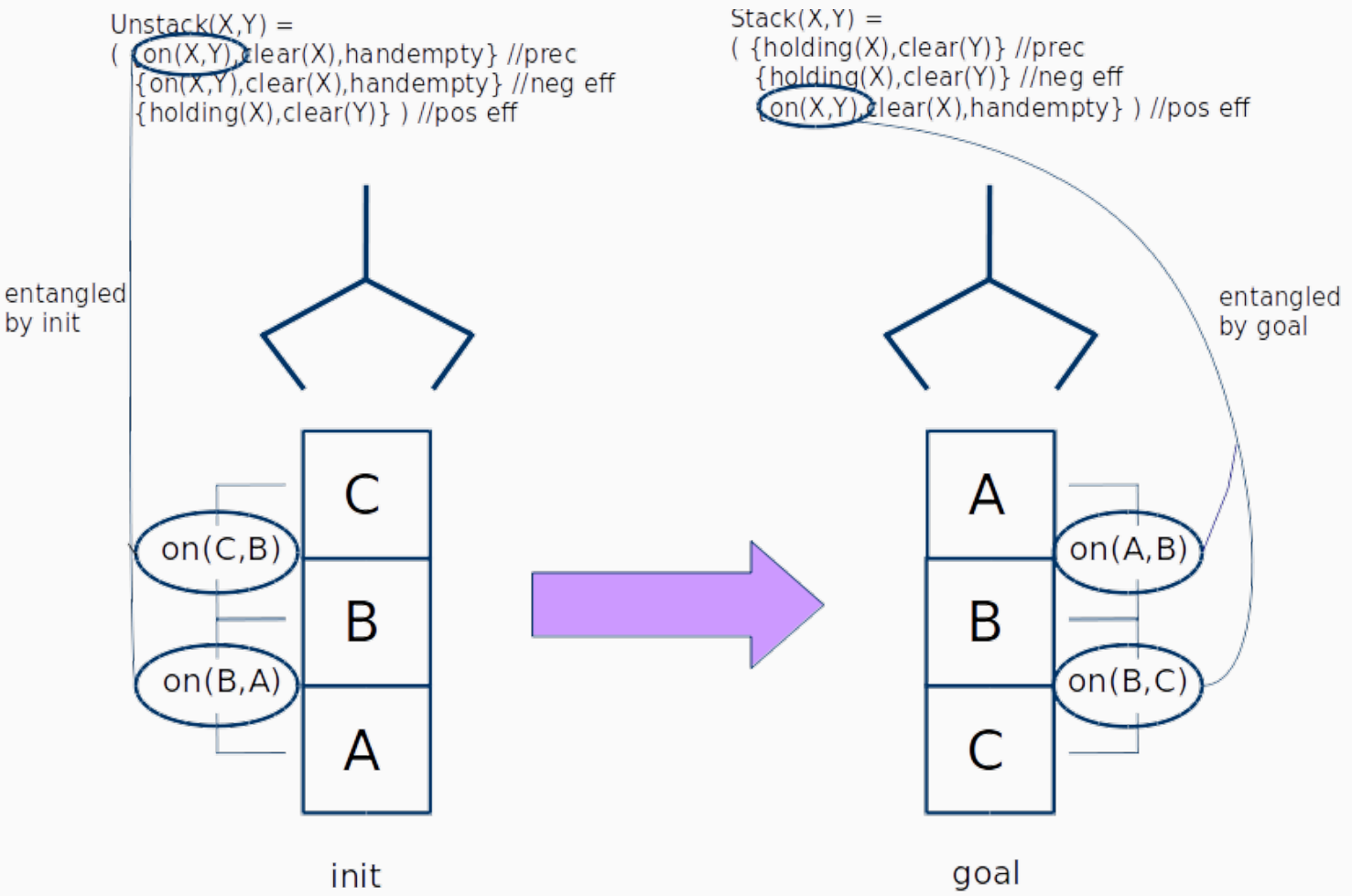
Entaglements

- Relations between planning operators and predicates [9, 11]
- Aim at reducing Branching Factor
- Often domain-specific (rather than problem-specific)
- Can be encoded into the domain models and problem specifications
- Can be learnt from training plans, solutions of simple planning tasks

Outer Entanglements

- Relations between planning operators and initial or goal predicates [9]
- Aim at reducing the number of instances of planning operators
- An **Entanglement by init** allows only instances of a given operator that require instances of a given predicate present in an initial state
- An **Entanglement by goal** allows only instances of a given operator that achieve instances of a given predicate present in a goal

Outer Entanglements - example



Outer Entanglements - encoding

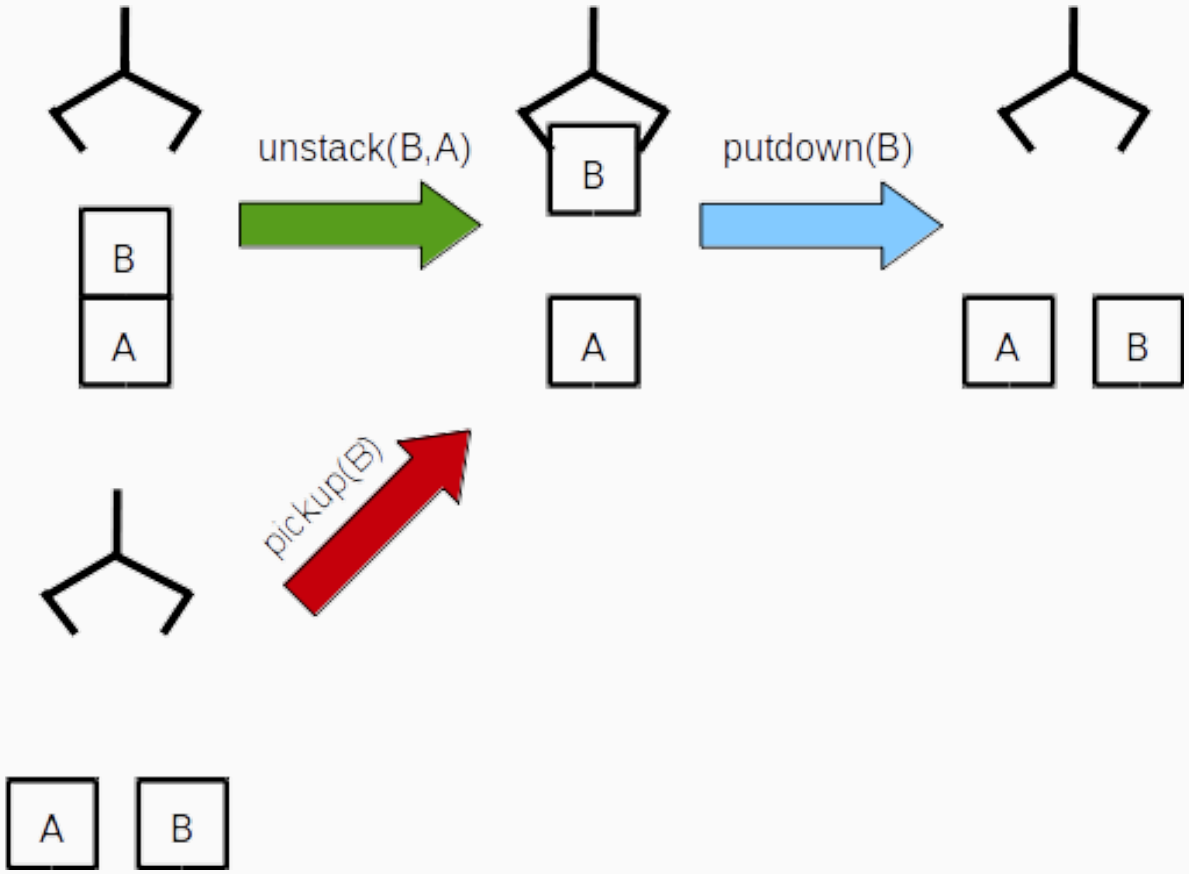
Supplementary static predicates are used to filter out undesirable operators' instances

```
 (:action unstack
   :parameters ( ?x - block ?y - block)
   :precondition (and (clear ?x)(handempty)
                      (on ?x ?y)(stai_on ?x ?y))
   :effect (and (holding ?x)(clear ?y)
                (not (clear ?x))(not (handempty))
                (not (on ?x ?y)))
)
```

Inner Entanglements

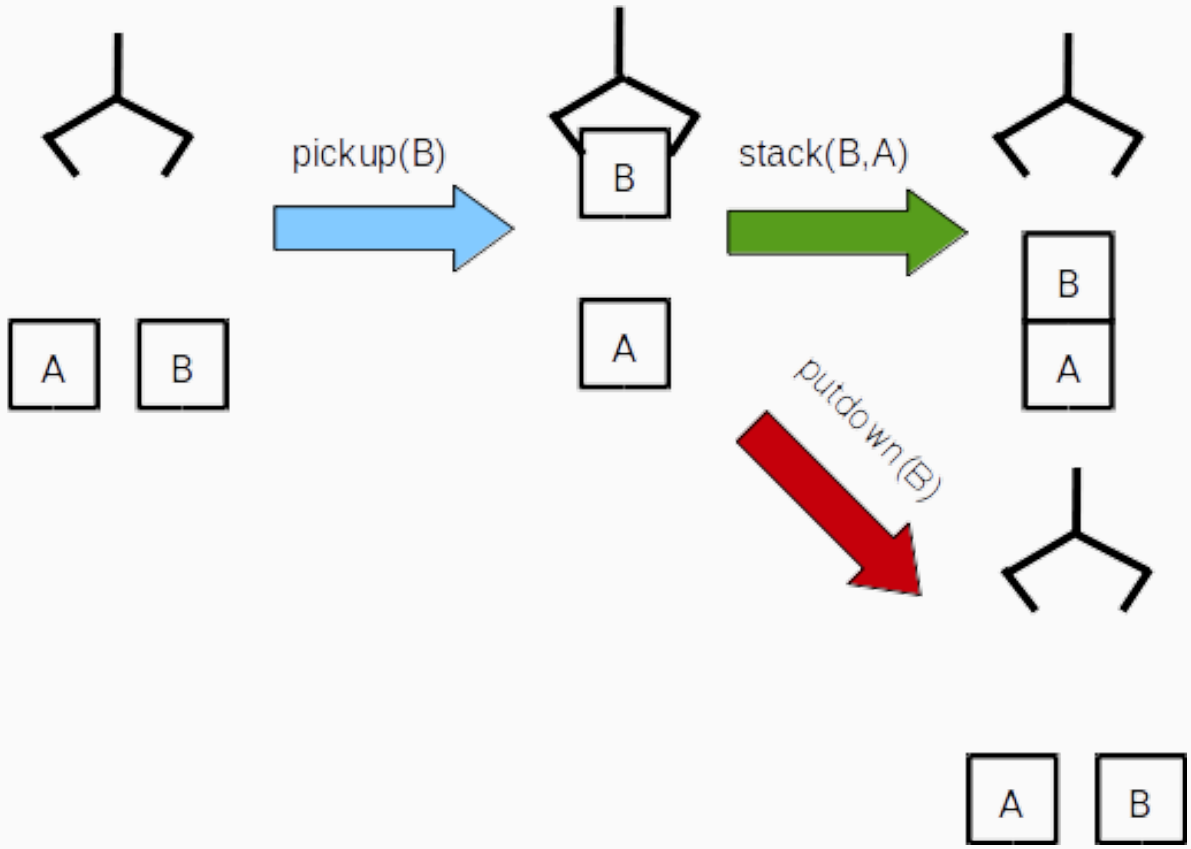
- Inner entanglements are relations of exclusivity of predicate “requirement” or “achievement” between pairs of planning operators [11]
- An **Entanglement by preceding** – A given predicate for a given operator is exclusively achieved by another operator
- An **Entanglement by succeeding** – An operator achieves a given predicate exclusively for a given operator

Entanglements by preceding - example



`putdown(?x)` is **entangled by preceding** `unstack(?x ?y)` with `(holding ?x)`

Entanglements by succeeding - example



`pickup(?x)` is **entangled by succeeding** `stack(?x ?y)` with `(holding ?x)`

Inner Entanglements - encoding

Supplementary predicates are used to “lock out” undesirable operators’ sequences

```
 (: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)(not (pick-up_stack_succ_holding ?x)))
)

(:action put-down
  :parameters ( ?x - block)
  :precondition (and (holding ?x)(pick-up_stack_succ_holding ?x))
  :effect (and (not (holding ?x))(clear ?x)(handempty)(ontable ?x))
)

(:action stack
  :parameters ( ?x - block ?y - block)
  :precondition (and (holding ?x)(clear ?y))
  :effect (and (on ?x ?y)(not (clear ?y))(clear ?x)(handempty)
               (not (holding ?x))(pick-up_stack_succ_holding ?x)))

)
```

Entanglements - benefits and drawbacks

+ Pruning unpromising alternatives in the search space
+ Memory requirements (outer entanglements even reduce memory requirements)
+ Easy to learn

- The learning approach might learn incorrect entanglements
- Utility problem (especially for inner entanglements)

The ASAP planner

- The ASAP planner learns the most promising couple (encoding,planner) per domain [39]
- Considered encodings: Original, Macros, Outer Entanglements, Inner Entanglements, Both entanglements
- Considered planners: FF, SatPlan, Mp, Lama, LPG, SGPlan, Probe

ASAP - selected couples per domain

Domain	ASAPs	ASAPq
Barman	⟨Outer, SGPlan⟩	⟨Original, SGPlan⟩
BlocksWorld	⟨Both, FF⟩	⟨Both, FF⟩
Depots	⟨Outer, FF⟩	⟨Both, LPG⟩
Gripper	⟨Outer, SGPlan⟩	⟨Outer, Mp⟩
Gold Miner	⟨Macro, FF⟩	⟨Both, SatPlan⟩
Matching-BW	⟨Macro, LPG⟩	⟨Outer, LPG⟩
Parking	⟨Original, FF⟩	⟨Inner, Lama⟩
Rovers	⟨Macro, LPG⟩	⟨Macro, Lama⟩
Satellite	⟨Outer, LPG⟩	⟨Outer, LPG⟩
Spanner	⟨Outer, Mp⟩	⟨Original, LPG⟩
TPP	⟨Both, LPG⟩	⟨Outer, Lama⟩

Table 1: For every domain, the couple selected by ASAPs and ASAPq.

ASAP vs. best “basic” planner

Domain	BestS	Time IPC		BestQ	Quality IPC	
		ASAPs	BestS		ASAPq	BestQ
Barman	SGPlan	23.1	**28.1**	SGPlan	30.0	30.0
BW	Probe	**30.0**	5.8	Probe	**29.5**	18.1
Depots	Probe	**30.0**	8.7	Probe	**30.0**	26.2
Gripper	LPG	**30.0**	5.0	LPG	**30.0**	8.3
Gold-m	Mp	**30.0**	16.5	LPG	**30.0**	29.9
M-BW	Lama	**30.0**	8.7	SatPlan	**26.4**	20.4
Parking	FF	5.0	5.0	FF	3.8	**4.6**
Rovers	LPG	**28.0**	22.0	LPG	**29.3**	26.1
Satellite	LPG	**29.0**	27.8	LPG	28.8	**29.7**
Spanner	LPG	10.6	**25.6**	LPG	30.0	30.0
TPP	Lama	**20.0**	3.4	Lama	**30.0**	9.1
All above		**265.7**	156.6		**297.8**	232.4

Table 2: Time/quality IPC score (max score 30 per domain) by ASAP and the best planner in the original encodings for the selected domains.

MUM

MUM [12] combines macros with outer entanglements

- Outer entanglements **reduce the branching factor** macros might introduce
- Outer entanglements serve as **heuristics** for macro generation
- Incompleteness issues of the outer entanglements learning process is alleviated by applying them only on macros

MUM - results

	Coverage			Fastest			IPC Score		
	O	W	M	O	W	M	O	W	M
FF	1	2	**36**	0	0	**36**	0.5	1.1	**36.0**
LAMA	39	13	**116**	5	0	**113**	30.4	10.8	**115.7**
LPG	**99**	30	86	**89**	30	24	**97.9**	30.0	67.1
Mp	8	33	**41**	2	9	**41**	6.9	30.6	**41.0**
Probe	80	54	**86**	20	0	**75**	64.4	38.1	**84.3**
Total	227	132	**365**	116	39	**289**	199.1	110.6	**344.1**

Table 3: Cumulative results across the IPC-7 learning track domains. O – original, W – Wizard [32], M - MUM

Learning Planner-independent DK - summary

DK such as macros or entanglements is easy to learn

- Usually, several simple training instances are sufficient
- No explicit knowledge from an engineer is needed

DK such as macros or entanglements generalizes well

DK often provides performance improvement across domains and planners

Various impact per domain and planner

Manually Specified DK

Can be (much more) efficient than learnt DK

Needs an expert to encode it

Planner-independent types of DK include:

- Procedural DCK (Domain Control Knowledge) [4] (can be also learnt to some extent [1])
- Transition-based DCK (Domain Control Knowledge) [10]

Procedural DCK

- Follows the Procedural Programming paradigm
- Action-centric (in contrast to LTL-based DCK)
- Describe how plans should be generated
- Can be encoded in planning domain models

Procedural DCK - Examples [4]

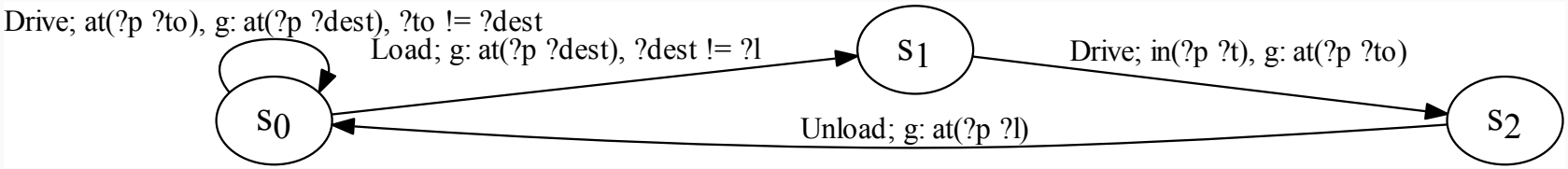
- **while** $\neg$`clear(B)` **do** π`(b-block)putOnTable(b)` : while `B` is not clear choose any `b` of type block and put it on the table.
- **any***;`loaded(A,Truck)?` : Perform any sequence of actions until `A` is loaded in `Truck`.
- `(load(C,P);fly(P,LA)` | `load(C,T);drive(T,LA))` : Either load `C` on the plane `P` or on the truck `T`, and perform the right action to move the vehicle to `LA`.

Transition-based DCK

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

Transition-based DCK - Example

1. An empty truck (can carry at most one package) should move only to locations where some package is waiting to be delivered
2. 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



Transition-based DCK - Results

Planner	Barman		CaveDiving		ChildSnack		CityCar		Hiking		Nomystery	
	O	E	O	E	O	E	O	E	O	E	O	E
Lama	16	**20**	6	7	0	19	5	**20**	5	19	14	14
Mercury	7	**20**	2	3	5	20	2	**20**	8	17	13	15
MpC	0	**20**	4	4	7	20	9	**20**	7	3	6	5
Probe	18	**20**	1	7	0	15	8	**20**	13	19	5	11
Yahsp	0	**20**	N/A	N/A	0	N/A	N/A	N/A	13	10	8	12
Bfs-f	20	**20**	7	7	8	8	5	**20**	2	14	14	15

Table 4: Coverage for original (O) and enhanced (E) domain models.

Noticeably, in some domains (e.g. Barman and CityCar)
Transition-based DCK “determinizes” the planning process

Acquiring and Using DK

Simpler kinds of DK (e.g. macros or entanglements)

+ Easy to learn
+ No planning expertise needed
- Not always efficient

“Sophisticated” kinds of DK (e.g. procedural or transition-based DCK)

+ In some cases can “determinize” the planning process
+ The performance of domain-independent planner can be close to domain-dependent ones
- Difficult to learn
- Engineers should have required domain-specific knowledge

Domain Model Configuration

The idea [40]: given a PDDL model, configure the order in which:

- domain predicates are declared
- operators are listed
- within every operator, the order of pre and post conditions

To each configurable element is associated a continuous value in the interval [0, 1]. Elements are ordered by their precedence. The well-known SMAC [22] tool has been used for the configuration.

Domain Model Configuration: Example

Operator: ($pre1$, $pre2$, $pre3$) ($post1$, $post2$, $post3$, $post4$)

7 parameters: $p1$, $p2$, $p3$ (pre-conditions) and $p4$, $p5$, $p6$, $p7$ (post-conditions).

SMAC searches in the parameter space $[0, 1]^7$.

Domain Model Configuration: Example

Operator: ($pre1$, $pre2$, $pre3$) ($post1$, $post2$, $post3$, $post4$)

7 parameters: $p1$, $p2$, $p3$ (pre-conditions) and $p4$, $p5$, $p6$, $p7$ (post-conditions).

SMAC searches in the parameter space $[0, 1]^7$.

If SMAC evaluates, e.g., [$p1$=0.32, $p2$=0.001, $p3$=0.7, $p4$=0.98, $p5$=0.11, $p6$=0.34, $p7$=0.35]

Operator: ($pre2$, $pre1$, $pre3$), and ($post2$, $post3$, $post4$, $post1$).

Domain Model Configuration: Surprise Me

Interesting results:

Domain Model Configuration: Surprise Me

Interesting results:

- Can we improve the performance of a given planner? YES

Domain Model Configuration: Surprise Me

Interesting results:

- Can we improve the performance of a given planner? YES
- Does configuration affect relative performance of planners? YES

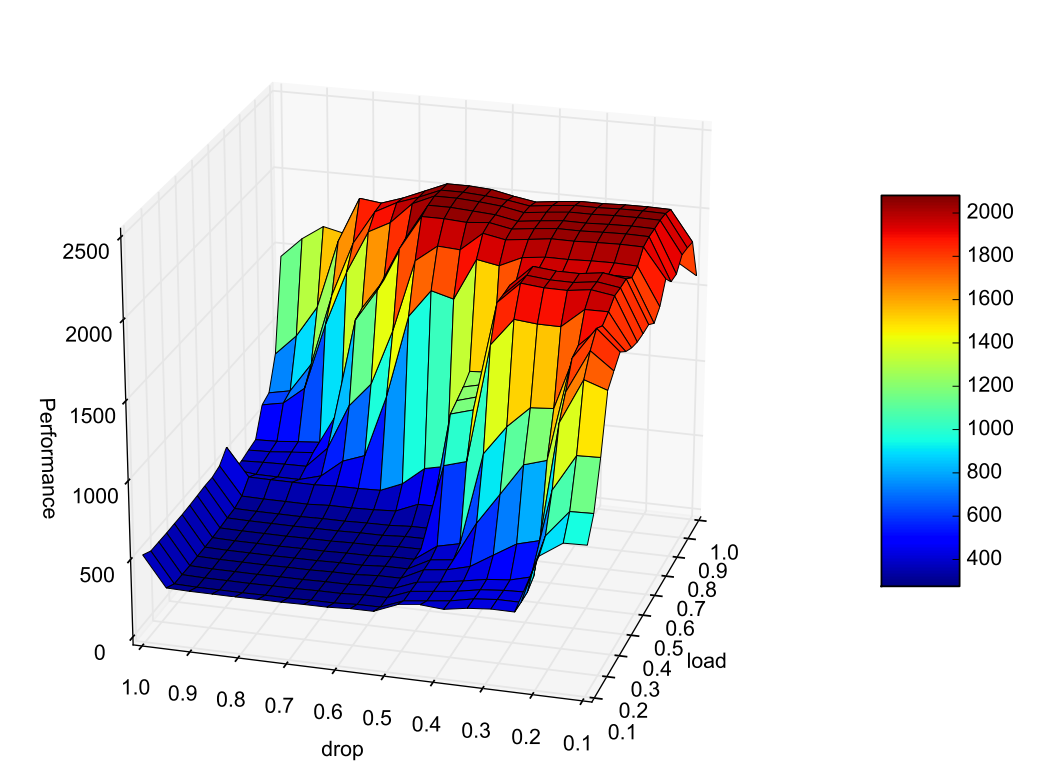
Domain Model Configuration: Surprise Me

Interesting results:

- Can we improve the performance of a given planner? YES
- Does configuration affect relative performance of planners? YES
- Is there a configuration that generally improves performance?

	IPC score		PAR10		Solved	
Planner	B	O	B	O	B	O
Jasper	**252.2**	195.9	**975.3**	1375.9	**69.4**	56.3
LPG	**261.2**	192.1	**826.0**	926.2	**72.9**	69.7
Mp	**232.5**	167.9	**1109.4**	1637.5	**63.8**	46.1
Mercury	**236.8**	183.9	**1126.6**	1381.5	**63.5**	55.8
Probe	**294.2**	205.5	**602.3**	1265.3	**81.0**	58.7
Yahsp3	**310.9**	260.0	**310.7**	390.6	**90.1**	87.4

Domain Model Configuration: Example



The average PAR10 performance of Yahsp in the Depots domain, as a function of the values of the parameters corresponding to the load and drop operators. The lower the PAR10 value, the better the performance.

Domain Model Configuration: Take-Home Message

- Domain model configuration strongly affect planners' performance
  - It is also possible to (significantly) decrease performance

Domain Model Configuration: Take-Home Message

- Domain model configuration strongly affect planners' performance
  - It is also possible to (significantly) decrease performance

- General observations:
  - operators should be listed according to *directionality* or frequency
  - preconditions most likely to be unsatisfied should be listed earlier
  - main effects should be listed earlier
  - predicate should be declared following their frequency as operators' preconditions

On the Boundaries

Automatic Domain Model Acquisition

Idea: since domain model encoding is hard for humans, can we do it in an automated way?

Automatic Domain Model Acquisition

Idea: since domain model encoding is hard for humans, can we do it in an automated way?

In short: **Yes**.

Automatic Domain Model Acquisition

Idea: since domain model encoding is hard for humans, can we do it in an automated way?

In short: **Yes**.

Approaches vary with regards to a number of dimensions, such as:

- Input requirements
- Model provided as output
- Supported languages
- Ability to deal with noise

Automatic Domain Model Acquisition: Approaches

Another by no mean complete list of tools, following [24].

Criteria	AMAN [48]	ARMS [44]	LOCM [13]	LSO-NIO [31]	Opmaker [28]	RIM [49]	SLAF [36]
Input	NP	BK,P	P	PDM,NP	PDM,P	PDM,P	Pr,IS
Output	DM	DM	DM	DM	DM,H	RDM	DM
Language	STRIPS	PDDL	PDDL	STRIPS	OCL	STRIPS	SL
Noise	Y	N	N	Y	N	N	N

P: Plan traces; BK: Background Knowledge; PDM: Partial Domain Model; Pr: Predicates; IS: Intermediate States; NP: Noisy Plans; DM: Domain Model; RDM: Refined Domain Model; H: Heuristics.

There is also a range of tools building on top of LOCM, such as LOP [20] or ASCOL [23].

Crowdsourced Domain Model Acquisition

Assumption: not many plan traces (or similar source of information) but many *annotators*.

Crowdsourced Domain Model Acquisition

Assumption: not many plan traces (or similar source of information) but many *annotators*.

CAMA (Crowdsourced Action-Model Acquisition) [47] exploits this approach. Given a set of operators (names and parameters, only):

- A set of annotators is used for accepting / rejecting potential pre and post conditions
- Plan traces, whether available, are used as further data
- a MAXSAT solver, given a formalisation of the previous data under the form of soft constraints, is used for generating the final model.

Results are encouraging!

Conclusions

Summary and Conclusions

- The Domain model is the corner stone of any planning application (or, in general, use of planning).
- *stretched* between two almost antithetic perspectives: planning engines and humans.
- Decoupling domain models and (manually specified) Domain Knowledge provides a good trade-off between both perspectives.
- Automatic extraction of Domain Knowledge, under the form of reformulation, also supports both perspectives.
- Little attention has been given to KE of models.

Challenges

(Take this slide as food for thought)

- How can we better handle models evolution and validation?

Challenges

(Take this slide as food for thought)

- How can we better handle models evolution and validation?
- Can we provide a notion of *quality* of models? Is it possible to assess quality statically, or is it pivotal to dynamically check it?

Challenges

(Take this slide as food for thought)

- How can we better handle models evolution and validation?
- Can we provide a notion of *quality* of models? Is it possible to assess quality statically, or is it pivotal to dynamically check it?
- Planning is being exploited in complex applications: we must support cooperation in KE tools for planning

Challenges

(Take this slide as food for thought)

- How can we better handle models evolution and validation?
- Can we provide a notion of *quality* of models? Is it possible to assess quality statically, or is it pivotal to dynamically check it?
- Planning is being exploited in complex applications: we must support cooperation in KE tools for planning
- From OOP to Planning: Design patterns. (and definition of alternatives that can be explored using DK..)

Thank you!

(That’s all Folks!)

References I

[1] J. S. Aguas, S. J. Celorrio, and A. Jonsson.
Generalized planning with procedural domain control knowledge.
In *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling, ICAPS 2016, London, UK, June 12-17, 2016.*, pages 285–293, 2016.

[2] C. Areces, F. Bustos, M. A. Dominguez, and J. Hoffmann.
Optimizing planning domains by automatic action schema splitting.
In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*, 2014.

References II

[3] M. Asai and A. Fukunaga.
Solving large-scale planning problems by decomposition and macro generation.
In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS 2015, Jerusalem, Israel, June 7-11, 2015.*, pages 16–24, 2015.

[4] J. A. Baier, C. Fritz, and S. A. McIlraith.
Exploiting procedural domain control knowledge in state-of-the-art planners.
In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS 2007, Providence, Rhode Island, USA, September 22-26, 2007*, pages 26–33, 2007.

References III

[5] J. Barreiro, M. Boyce, M. Do, J. Frank, M. Iatauro, T. Kichkaylo, P. Morris, J. Ong, E. Remolina, T. Smith, et al.
Europa: A platform for ai planning, scheduling, constraint programming, and optimization.
In *Proc. of the International Competition on Knowledge Engineering for Planning and Scheduling*, 2012.

[6] A. Botea, M. Enzenberger, M. Müller, and J. Schaeffer.
Macro-ff: Improving AI planning with automatically learned macro-operators.
J. Artif. Intell. Res. (JAIR), 24:581–621, 2005.

References IV

[7] B. Bredeweg, P. Salles, A. Bouwer, J. Liem, T. Nuttle, E. Cioaca, E. Nakova, R. Noble, A. L. R. Caldas, Y. Uzunov, et al.
Towards a structured approach to building qualitative reasoning models and simulations.
Ecological Informatics, 3(1):1–12, 2008.

[8] L. Chrpa.
Generation of macro-operators via investigation of action dependencies in plans.
Knowledge Eng. Review, 25(3):281–297, 2010.

References V

[9] L. Chrpa and R. Barták.
Reformulating planning problems by eliminating unpromising actions.
In *Eighth Symposium on Abstraction, Reformulation, and Approximation, SARA 2009, Lake Arrowhead, California, USA, 8-10 August 2009*, 2009.

[10] L. Chrpa and R. Barták.
Guiding planning engines by transition-based domain control knowledge.
In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016.*, pages 545–548, 2016.

References VI

[11] L. Chrpa and T. L. McCluskey.
On exploiting structures of classical planning problems: Generalizing entanglements.
In *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, pages 240–245, 2012.

[12] L. Chrpa, M. Vallati, and T. L. McCluskey.
MUM: A technique for maximising the utility of macro-operators by constrained generation and use.
In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*, 2014.

References VII

[13] S. N. Cresswell, T. L. McCluskey, and M. M. West.
Acquiring planning domain models using locm.
The Knowledge Engineering Review, 28(02):195–213, 2013.

[14] M. D. Feblowitz, A. V. Riabov, and O. Udrea.
Planning-based composition of stream processing applications.
In *In ICAPS-12 System Demonstrations and Exhibits Track*, 2012.

[15] M. Fox and D. Long.
PDDL2.1: an extension to PDDL for expressing temporal planning domains.
J. Artif. Intell. Res. (JAIR), 20:61–124, 2003.

References VIII

[16] M. Fox and D. Long.
Modelling mixed discrete-continuous domains for planning.
J. Artif. Intell. Res. (JAIR), 27:235–297, 2006.

[17] J. Frank and A. Jónsson.
Constraint-based attribute and interval planning.
Constraints, 8(4):339–364, 2003.

[18] A. Gerevini, P. Haslum, D. Long, A. Saetti, and Y. Dimopoulos.
Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners.
Artif. Intell., 173(5-6):619–668, 2009.

References IX

[19] A. González-Ferrer, J. Fernández-Olivares, and L. Castillo.
JABBAH: A java application framework for the translation between business process models and htn.
In *Proc. of the International Competition on Knowledge Engineering for Planning and Scheduling*, 2009.

[20] P. Gregory and S. Cresswell.
Domain model acquisition in the presence of static relations in the LOP system.
In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI*, pages 4160–4164, 2016.

References X

[21] P. Haslum and P. Jonsson.
Planning with reduced operator sets.
In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems, Breckenridge, CO, USA, April 14-17, 2000*, pages 150–158, 2000.

[22] F. Hutter, H. H. Hoos, and K. Leyton-Brown.
Sequential model-based optimization for general algorithm configuration.
In *Proceedings of the 5th Learning and Intelligent OptimizatioN Conference (LION)*, pages 507–523, 2011.

[23] R. Jilani, A. Crampton, D. E. Kitchin, and M. Vallati.
Ascol: A tool for improving automatic planning domain model acquisition.
In *Proc. of AI*IA*, pages 438–451, 2015.

References XI

[24] R. Jilani, D. Kitchin, and M. Vallati.
Knowledge engineering tools in planning.
In *Proc. of KEPS*, 2014.

[25] R. E. Korf.
Macro-operators: A weak method for learning.
Artif. Intell., 26(1):35–77, 1985.

[26] D. L. Kovacs.
A multi-agent extension of pddl 3.1.
In *Proceedings of the 3rd Workshop on the International Planning Competition (IPC), 22nd International Conference on Automated Planning and Scheduling (ICAPS-2012)*, pages 19–27. ICAPS, 2012.

References XII

[27] T. McCluskey and J. Porteous.
Engineering and compiling planning domain models to promote validity and efficiency.
Artificial Intelligence, 95(1):1 – 65, 1997.

[28] T. L. McCluskey, S. Cresswell, N. E. Richardson, and M. M. West.
Automated acquisition of action knowledge.
In *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART)*, pages 93–100, 2009.

[29] T. L. McCluskey, T. Vaquero, and M. Vallati.
Issues in planning domain model engineering.
In *PlanSIG*, 2015.

References XIII

[30] D. Mcdermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins.
PDDL - The Planning Domain Definition Language.
Technical report, CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.

[31] K. Mourão, L. S. Zettlemoyer, R. P. A. Petrick, and M. Steedman.
Learning strips operators from noisy and incomplete observations.
In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 614–623, 2012.

References XIV

[32] M. A. H. Newton, J. Levine, M. Fox, and D. Long.
Learning macro-actions for arbitrary planners and domains.
In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS 2007, Providence, Rhode Island, USA, September 22-26, 2007*, pages 256–263, 2007.

[33] T. Plch, M. Chomut, C. Brom, and R. Barták.
Inspect, edit and debug pddl documents: Simply and efficiently with pddl studio.
In *System Demonstration – ICAPS*, 2012.

[34] S. Sanner.
Relational dynamic influence diagram language (RDDL): Language description.
http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf, 2010.

References XV

[35] M. Shah, L. Chrpa, F. Jimoh, D. Kitchin, T. McCluskey, S. Parkinson, and M. Vallati.
Knowledge engineering tools in planning: State-of-the-art and future challenges.
In *Proc. of KEPS*, 2013.

[36] D. Shahaf and E. Amir.
Learning partially observable action schemas.
In *The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference (AAAI-06)*, pages 913–919, 2006.

[37] R. Simpson, D. E. Kitchin, and T. McCluskey.
Planning domain definition using gipo.
Knowledge Engineering Review, 22(2):117–134, 2007.

References XVI

[38] D. E. Smith, J. Frank, and W. Cushing.
The ANML language.
In *Proceedings of Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*, 2008.

[39] M. Vallati, L. Chrpa, and D. E. Kitchin.
ASAP: an automatic algorithm selection approach for planning.
International Journal on Artificial Intelligence Tools, 23(6), 2014.

[40] M. Vallati, F. Hutter, L. Chrpa, and T. L. McCluskey.
On the effective configuration of planning domain models.
In *Proc. of IJCAI*, pages 1704–1711, 2015.

References XVII

[41] T. S. Vaquero, R. Tonaco, G. Costa, F. Tonidandel, J. R. Silva, and J. C. Beck.
itSIMPLE4.0: Enhancing the modeling experience of planning problems.
In *System Demonstration – ICAPS*, 2012.

[42] J. Vodrážka and L. Chrpa.
Visual design of planning domains.
In *Proc. of KEPS*, 2010.

[43] G. Wickler, L. Chrpa, and T. McCluskey.
Creating planning domain models in kewi.
In *Proc. of KEPS*, 2014.

References XVIII

[44] Q. Yang, K. Wu, and Y. Jiang.
Learning action models from plan examples using weighted max-sat.
Artificial Intelligence, 171(2-3):107–143, Feb. 2007.

[45] H. L. Younes and M. L. Littman.
Ppddl 1.0: An extension to pddl for expressing planning domains with probabilistic effects.
Techn. Rep. CMU-CS-04-162, 2004.

[46] N. Zhou, R. Barták, and A. Dovier.
Planning as tabled logic programming.
TPLP, 15(4-5):543–558, 2015.

[47] H. Zhuo.
Crowdsourced action-model acquisition for planning.
In *AAAI Conference on Artificial Intelligence*, 2015.

References XIX

[48] H. H. Zhuo and S. Kambhampati.
Action-model acquisition from noisy plan traces.
In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI-13)*, pages 2444–2450. AAAI Press, 2013.

[49] H. H. Zhuo, T. Nguyen, and S. Kambhampati.
Refining incomplete planning domain models through plan traces.
In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pages 2451–2457, 2013.