Affordance-based reasoning for robot task planning

Iman Awaad
Gerhard K. Kraetzschmar

Joachim Hertzberg

Bonn-Rhein-Sieg University
and B-IT Center
Grantham-Allee 20
53757 Sankt Augustin, Germany

Osnabrück University
and DFKI RIC Osnabrück Branch
Albrechtstrasse 28
49076 Osnabrück, Germany

Hochschule Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

UNIVERSITÄT OSNABRÜCK

Affordances

Planning

Description Logics

\+

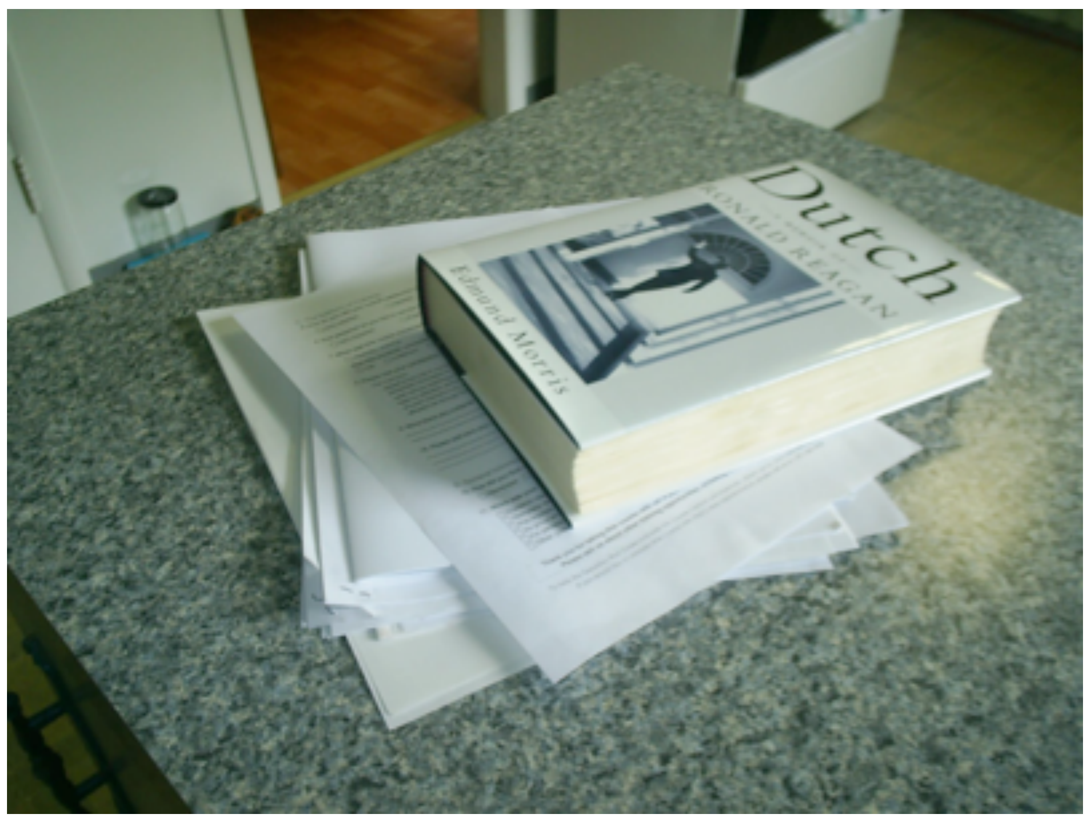
Enable robots to handle unexpected situations

RESEARCH GOAL

Enable robots to handle unexpected situations

Object substitution

Photo credit: http://blog.comfree.com/2013/05/03/clever-kitchen-storage-solutions/#.Uaekj-uDGJN



Enable robots to handle unexpected situations

Object substitution as tool usage

Enable robots to handle unexpected situations

Performance enhancement

Photo credit: http://www.instructables.com/id/Lazy-Line-Dry/step2/Clothes-with-plastic-hangers-How-to-do-it-fast-an/

Enable robots to handle unexpected situations

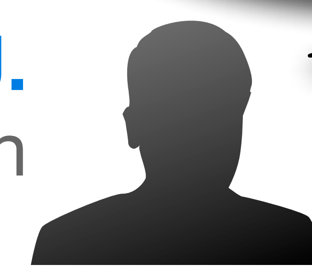
Action substitution

affordances

Affordances
are *opportunities for action* provided
by a particular object or environment

A closed door
does *not* afford *passage*!

James J.
Gibson



perceived affordances

Perceived affordances
allude to how an object may be interacted
with based on the *actor's goals*, *plans*,
values, *beliefs* and *past experiences*

It *might* afford
opening and *passage*!

Don
Norman

provide a means to represent
& use *a priori* knowledge

functional affordances

reduce the action space

Perceived affordances
allude to how an object may be interacted with based on the *actor's goals*, *plans*, *values*, *beliefs* and *past experiences*

It *might* afford
opening and *passage*!

Don
Norman

Enable intelligent behavior

handle underspecified commands

Show how to
model **Affordances** and use them
before **Planning**, during planning
+ **Description Logics** & at execution

Enable robots to handle unexpected situations

RESEARCH GOAL

real domains, especially in service robotics,
are really **hard** to model

1. **Model the domain**
2. Create the planning problem
3. Generate a plan
4. Execute/Monitor it

Model as **little** as possible

— could lose solutions

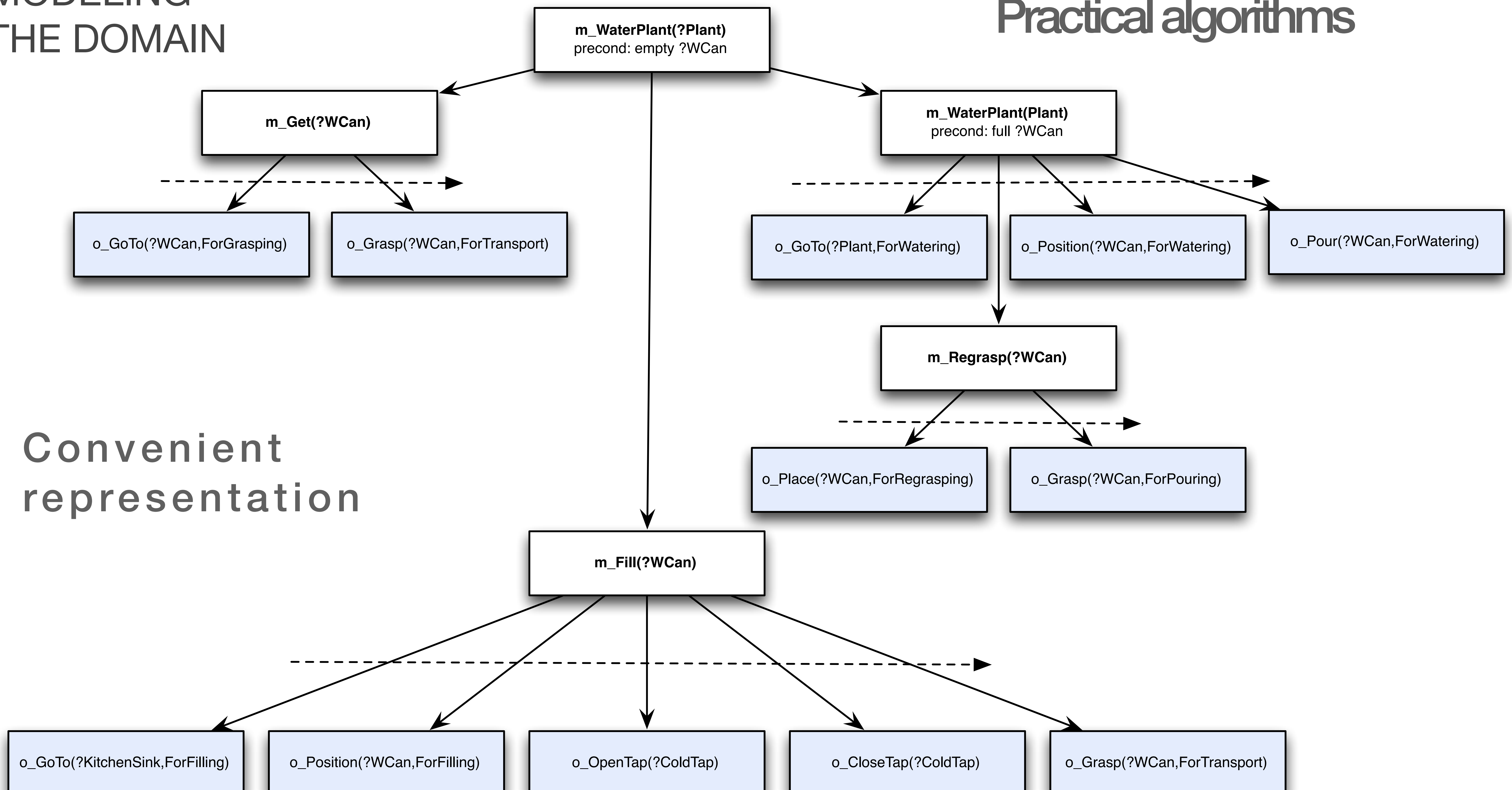
Use domain information to quickly solve hard problems

Model as **much** as possible

— difficult, time consuming

MODELING THE DOMAIN

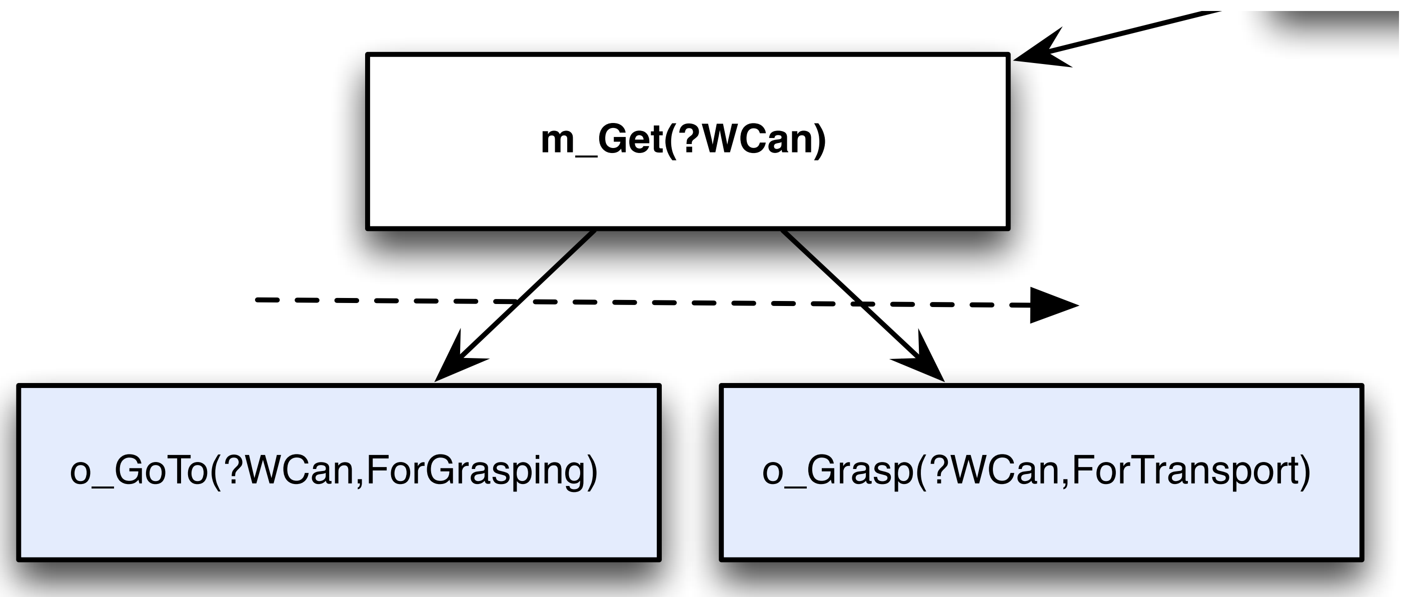
Practical algorithms



Convenient representation

[1] M. Ghallab, D. Nau, and P. Traverso. Automated planning: theory and practice. Morgan Kaufmann Publishers, Elsevier, 2004

MODELING THE DOMAIN



How do we know this is possible?

- Heterogenous hardware
- Faulty hardware

<table>
<tr><th>Construct</th><th>Syntax[1]</th><th colspan="5">Language[2]</th></tr>
<tr><td>Concept</td><td>A</td><td rowspan="4">$\mathcal{FL}_0$</td><td rowspan="7">$\mathcal{FL}^-$</td><td rowspan="8">$\mathcal{AL}$</td><td rowspan="11">$\mathcal{S}$</td><td rowspan="11">*</td></tr>
<tr><td>Role name</td><td>R</td></tr>
<tr><td>Intersection</td><td>$C \sqcap D$</td></tr>
<tr><td>Value restriction</td><td>$\forall R.C$</td></tr>
<tr><td>Limited existential quantification</td><td>$\exists R$</td><td rowspan="3"></td></tr>
<tr><td>Top or Universal</td><td>$\top$</td></tr>
<tr><td>Bottom</td><td>$\bot$</td></tr>
<tr><td>Atomic Negation</td><td>$\neg A$</td><td colspan="2"></td></tr>
<tr><td>Negation[3]</td><td>$\neg C$</td><td colspan="3">$\mathcal{C}$</td></tr>
<tr><td>Union</td><td>$C \sqcup D$</td><td colspan="3">$\mathcal{U}$</td></tr>
<tr><td>Existential restriction</td><td>$\exists R.C$</td><td colspan="3">$\mathcal{E}$</td></tr>
<tr><td>Number restrictions</td><td>$(\leq n\ R)\ (\geq n\ R)$</td><td colspan="2">$\mathcal{N}$</td><td>*</td><td colspan="2"></td></tr>
<tr><td>Nominals</td><td>$\{a_1 \ldots a_n\}$</td><td colspan="2">$\mathcal{O}$</td><td>*</td><td colspan="2"></td></tr>
<tr><td>Role hierarchy</td><td>$R \sqsubseteq S$</td><td colspan="2">$\mathcal{H}$</td><td>*</td><td colspan="2"></td></tr>
<tr><td>Inverse Role</td><td>R^-</td><td colspan="2">$\mathcal{I}$</td><td>*</td><td colspan="2"></td></tr>
<tr><td>Qualified number restriction</td><td>$(\leq n\ R.C)\ (\geq n\ R.C)$</td><td colspan="2">$\mathcal{Q}$</td><td></td><td colspan="2"></td></tr>
</table>

[1] A refers to atomic concepts, C and D refers to any concept definition, R refers to atomic roles and S refers to role definitions

[2] $\mathcal{FL}$ is used for structural DL languages and $\mathcal{AL}$ for attributive languages [BCM+03]. $\mathcal{S}$ is the name used for the language $\mathcal{ALC}_{R+}$, which is composed of $\mathcal{ALC}$ plus transitive roles.

[3] $\mathcal{ALC}$ and $\mathcal{ALCUE}$ are equivalent languages, since union ($\mathcal{U}$) and existential restriction ($\mathcal{E}$) can be represented using negation ($\mathcal{C}$).

Constructor	DL syntax	Example
`intersectionOf`	$C_1 \sqcap \ldots \sqcap C_n$	$Human \sqcap Male$
`unionOf`	$C_1 \sqcup \ldots \sqcup C_n$	$Doctor \sqcup Lawyer$
`complementOf`	$\neg C$	$\neg\ Male$
`oneOf`	$\{x_1 \ldots x_n\}$	{john,mary}
`allValuesFrom`	$\forall\ P.C$	$\forall\ hasChild.Doctor$
`someValuesFrom`	$\exists\ R.C$	$\exists\ hasChild.Lawyer$
`hasValue`	$\exists\ R.\{x\}$	$\exists\ citizenOf.\{\mathtt{USA}\}$
`minCardinality`	$(\geq n\ R)$	$(\geq 2\ hasChild)$
`maxCardinality`	$(\leq n\ R)$	$(\leq 1\ hasChild)$
`inverseOf`	R^-	$hasChild^-$

* Table reproduced from: Franz Baader, Ian Horrocks, and Ulrike Sattler. Description Logics. Handbook of Knowledge Representation, 2008.

MODELING THE DOMAIN

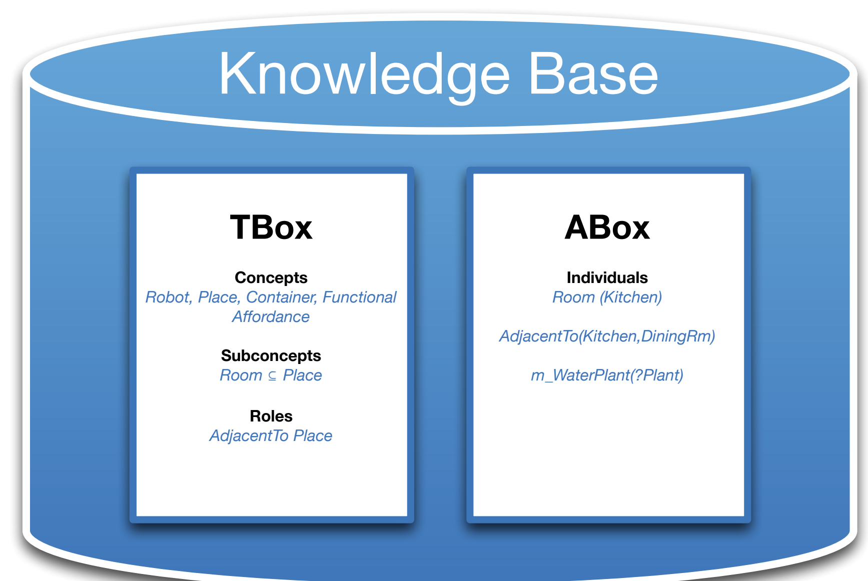
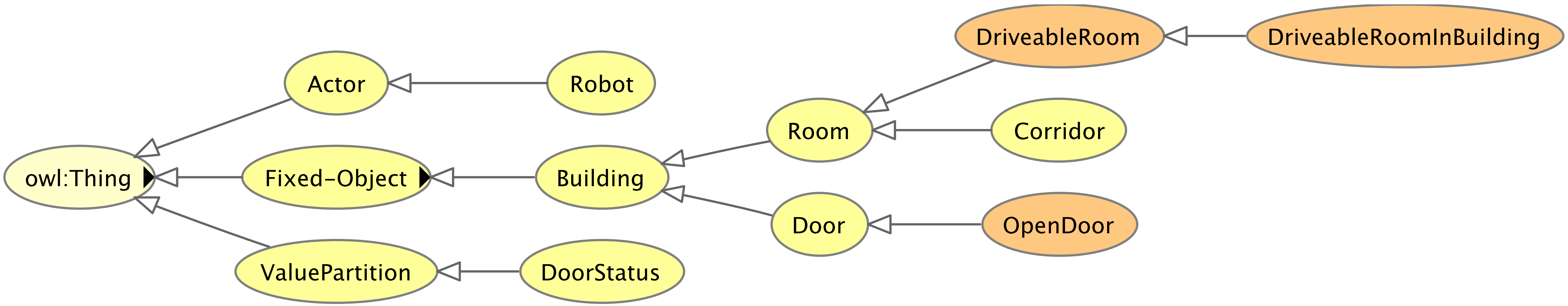
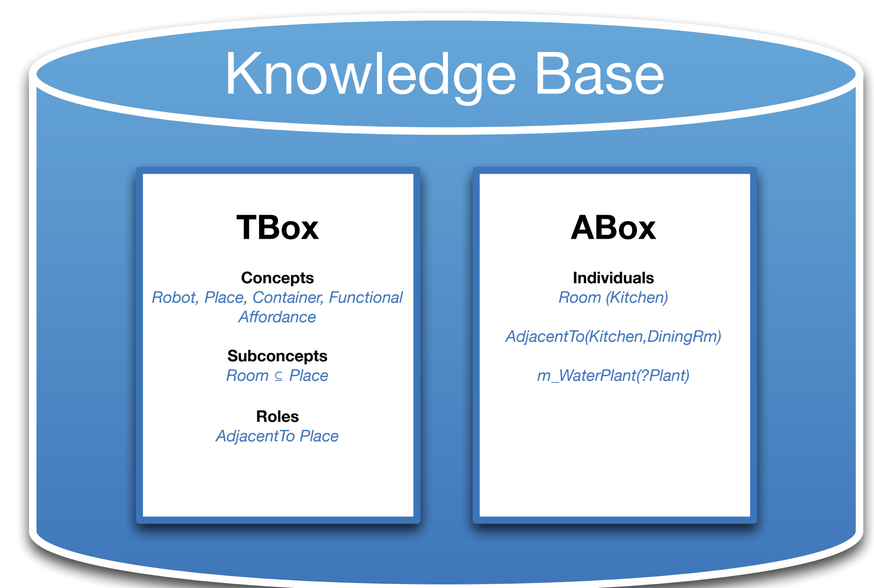


Table reproduced from R. Hartanto, Fusing DL Reasoning with HTN Planning as a Deliberative Layer in Mobile Robots. PhD thesis, University of Osnabrück, August 2009.





MODELING THE DOMAIN

Ontology reproduced from R. Hartanto, Fusing DL Reasoning with HTN Planning as a Deliberative Layer in Mobile Robots. PhD thesis, University of Osnabrück, August 2009.

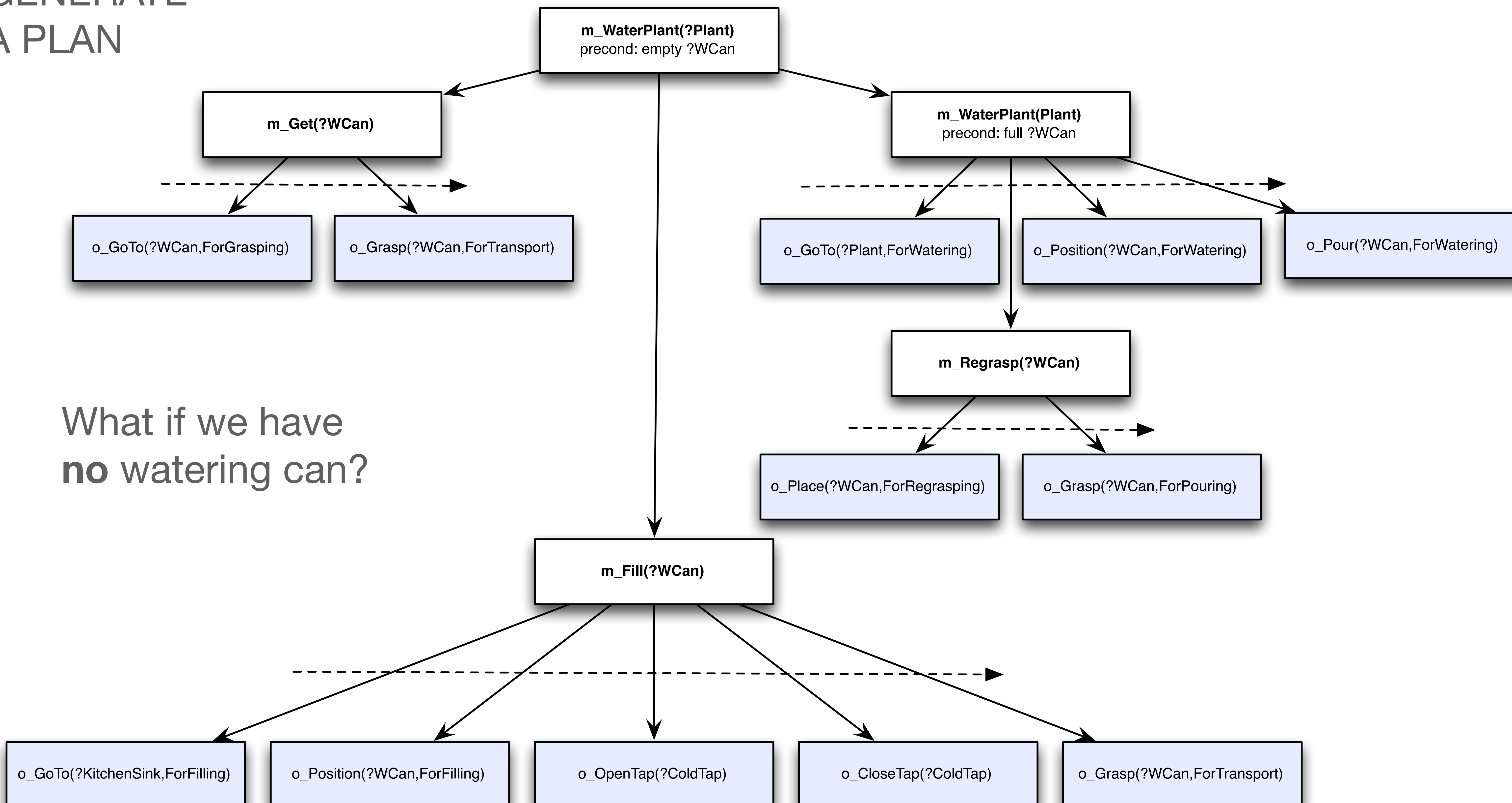
our initial state
is **HUGE**

1. Model the domain
2. **Create the planning problem**
3. Generate a plan
4. Execute/Monitor it

Use DL to infer relevant aspects of the domain

1. Model the domain
2. Create the planning problem
3. **Generate a plan**
4. Execute/Monitor it

GENERATE A PLAN

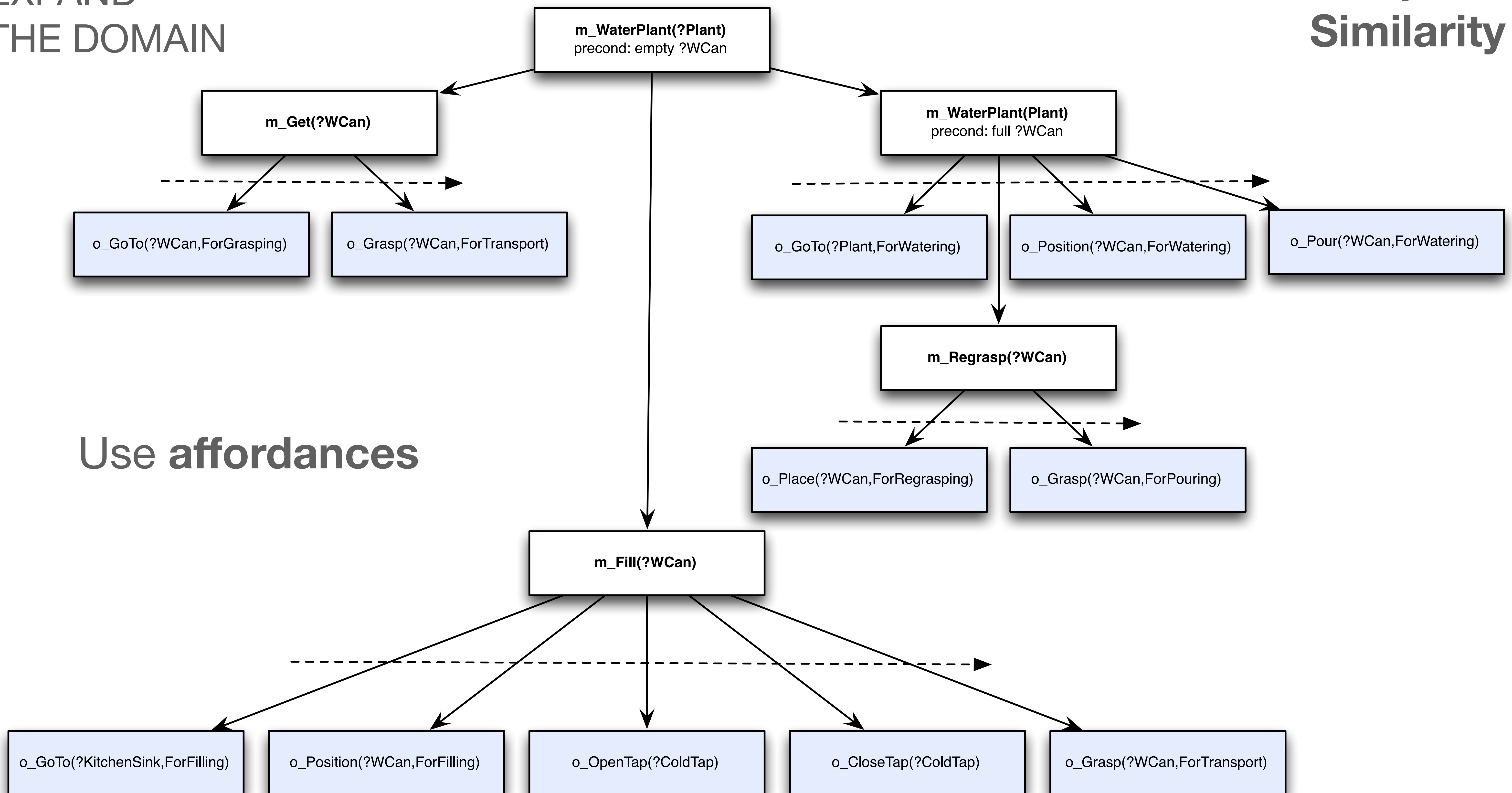


What if we have **no** watering can?

1. Model the domain
2. Create the planning problem
3. **Generate a plan**
4. Expand the domain and try again
5. Execute/Monitor it

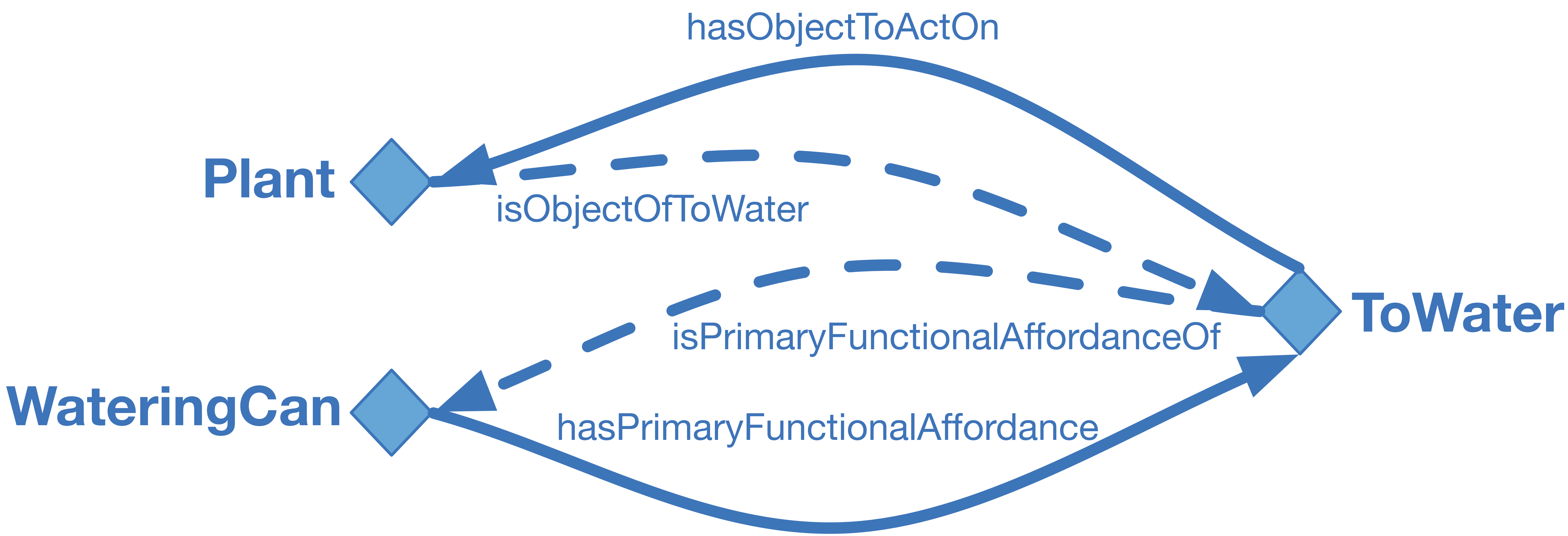
EXPAND
THE DOMAIN

...and **Conceptual Similarity**



Use **affordances**

FUNCTIONAL AFFORDANCES

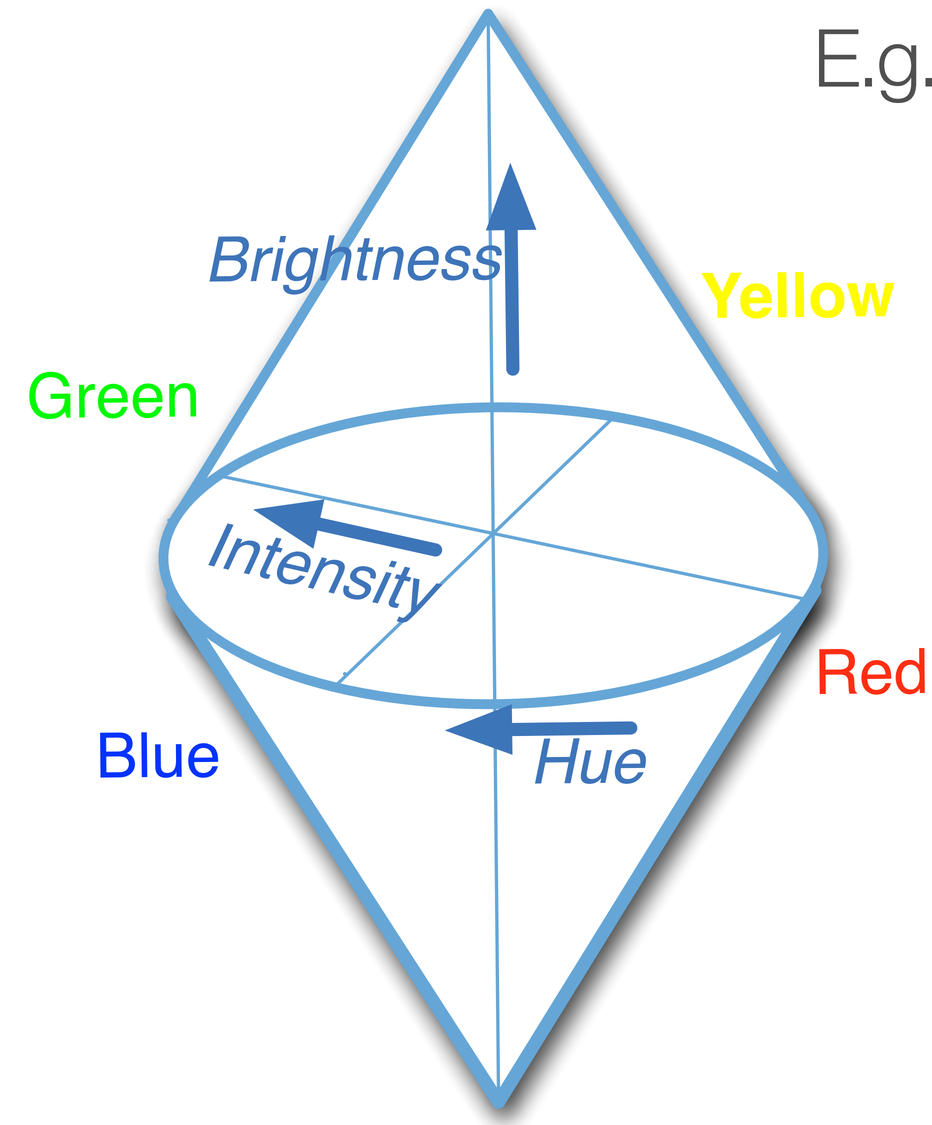


watering can |ˈwɔdərɪŋ ˌkæn|
noun
a portable water container with a long spout and a detachable perforated cap, used for watering plants.

Can we determine a relation between quality dimensions and given tasks?

E.g. Capacity to hold water; handle; spout

Conceptual spaces are composed by ***quality dimensions***



Multi-dimensional feature space:

- ***points*** denote objects
- ***regions*** denote concepts

CONCEPTUAL SIMILARITY

Gärdenfors, P., and Warglien, M. 2012. Using Conceptual Spaces to Model Actions and Events. *Journal of Semantics*.

Unique Instance

Common Instance

Same Functional Affordance
& Conceptually Similar

Same Functional Affordance

Conceptually Similar

Inferred Conceptual Similarity

E.g. only “*my_teacup*”

Increasing flexibility & Decreasing Constraints

Unique Instance

Common Instance

Same Functional Affordance
& Conceptually Similar

Same Functional Affordance

Conceptually Similar

Inferred Conceptual Similarity

E.g. closest instance of a “*teacup*”

Increasing flexibility & Decreasing Constraints

Unique Instance

Common Instance

Same Functional Affordance & Conceptually Similar

Same Functional Affordance

Conceptually Similar

Inferred Conceptual Similarity

E.g.closest object “*for drinking from*”, that matches “*small*, *bowl-shaped*, *container*, *handle*” (e.g. “mug”)

Increasing flexibility & Decreasing Constraints

Unique Instance

Common Instance

Same Functional Affordance
& Conceptually Similar

Same Functional Affordance

Conceptually Similar

Inferred Conceptual Similarity

E.g. closest object “*for drinking from*”
(e.g. “bottle”)

Increasing flexibility & Decreasing Constraints

Unique Instance

Common Instance

Same Functional Affordance
& Conceptually Similar

Same Functional Affordance

Conceptually Similar

Inferred Conceptual Similarity

E.g. “*small*, *bowl-shaped*,*container*, *handle*” (e.g. “measuring cup”)

Increasing flexibility & Decreasing Constraints

Unique Instance

Common Instance

Same Functional Affordance
& Conceptually Similar

Same Functional Affordance

Conceptually Similar

Inferred Conceptual Similarity

E.g. objects used "*for drinking from*" are usually "*small*, *cylindrical*, *container*, *glass*" (e.g. "jar")

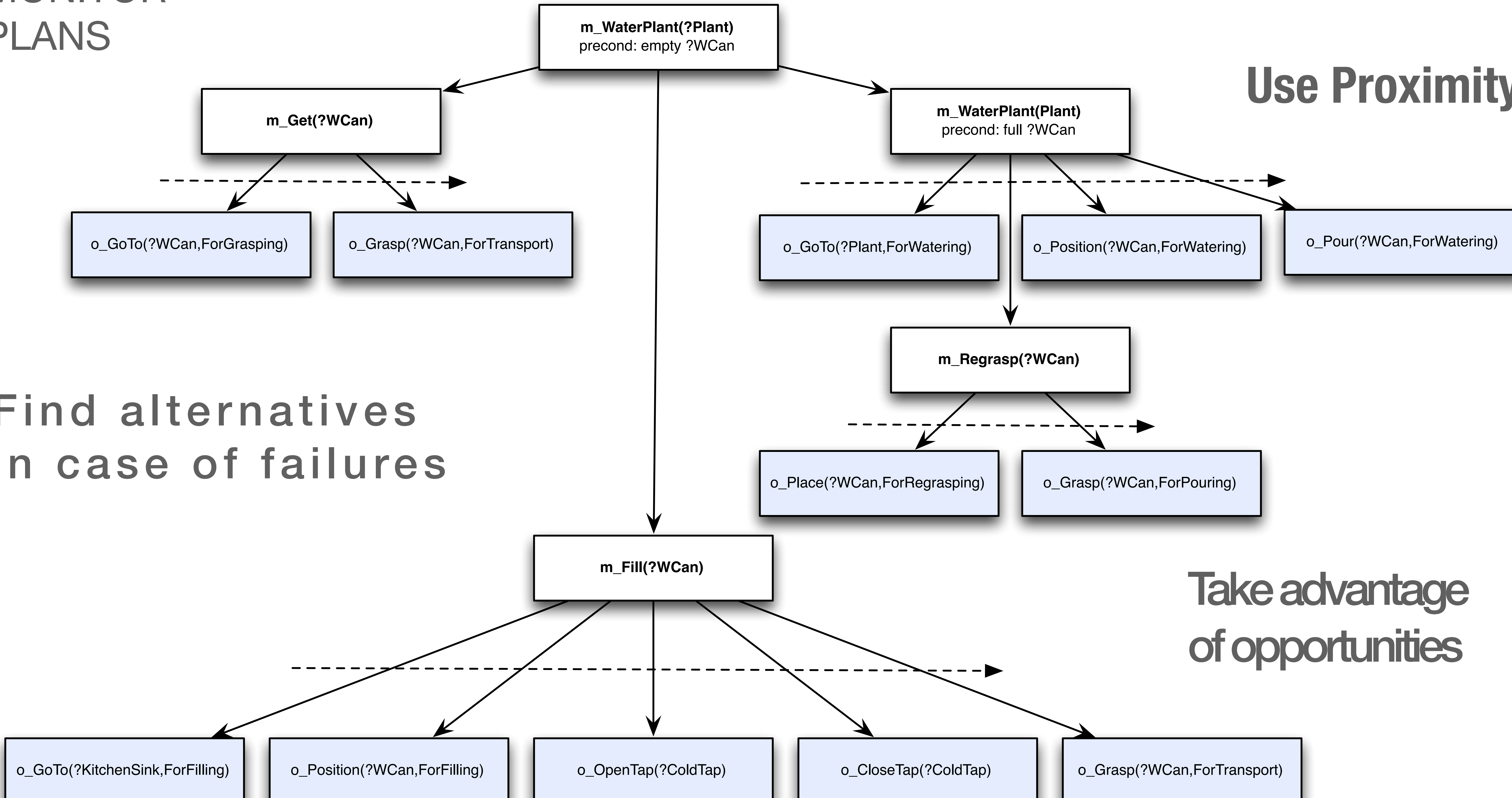
Increasing flexibility & Decreasing Constraints

incomplete information about
the environment

1. Model the domain
2. Create the planning problem
3. Generate a plan
4. **Execute/Monitor it**

Combine generated plans with action behaviors

EXECUTE/ MONITOR PLANS



Use Proximity

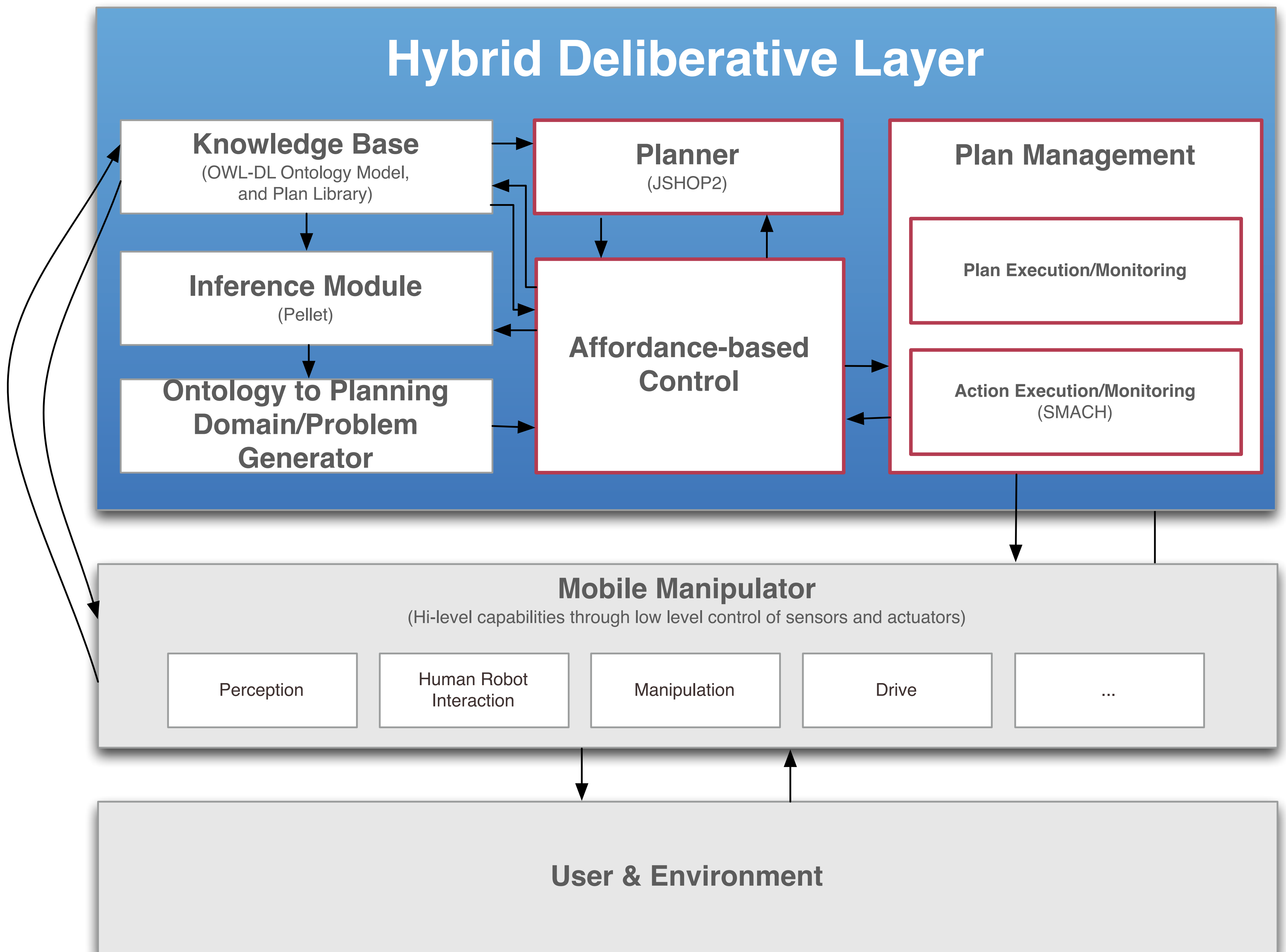
Find alternatives in case of failures

Take advantage of opportunities

USE ABSTRACT AFFORDANCES

1. Cluster behaviors by their effect on objects
2. Create one operator per cluster
3. Generate plans with these operators
4. Executed as the closest-matching behavior

To reduce complexity during planning



1. Receive command
2. Check plan library
3. Create planning problem
4. Generate Plan
5. Execute and monitor it

[1] R. Hartanto, Fusing DL Reasoning with HTN Planning as a Deliberative Layer in Mobile Robots. PhD thesis, University of Osnabrück, August 2009.

Architecture design

Proof of concept integrating planning with execution & monitoring

Integration into our b-it-bots RoboCup @Home framework

Modeling functional affordances in DL

Abstraction hierarchy for action substitution

Extend planner to lift over functional afffordances and use justification structures

Design the plan library (including preferences)

Test domain expansion phase

Extend this to enable action substitution

Enable instantiation of affordance behaviors at execution time using Conceptual Spaces

Enable object substitution as tool usage

Enable the performance enhancement use case

THANK YOU