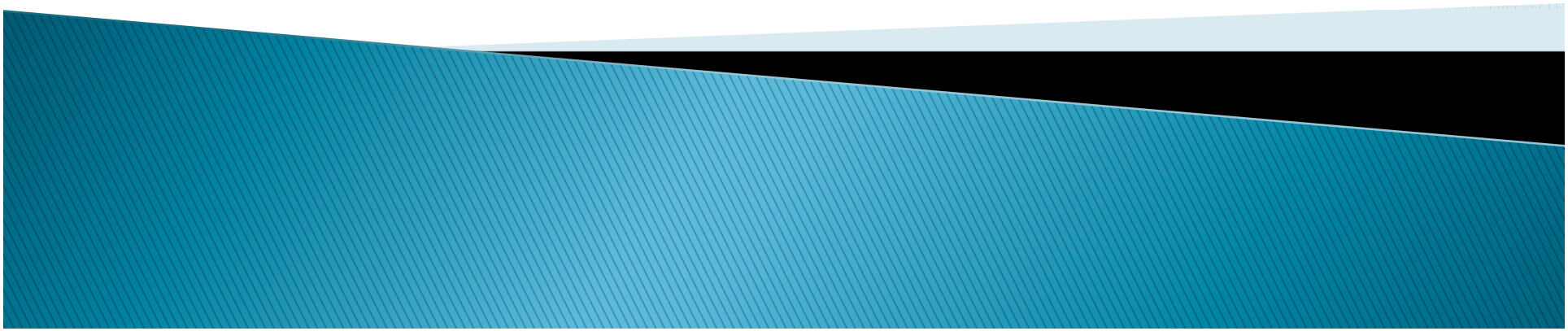
An Industrial Knowledge Representation for Kit Building Applications

S. Balakirsky, Z. Kootbally, C. Schlenoff, T. Kramer, S. Gupta

Presented by: Stephen Balakirsky



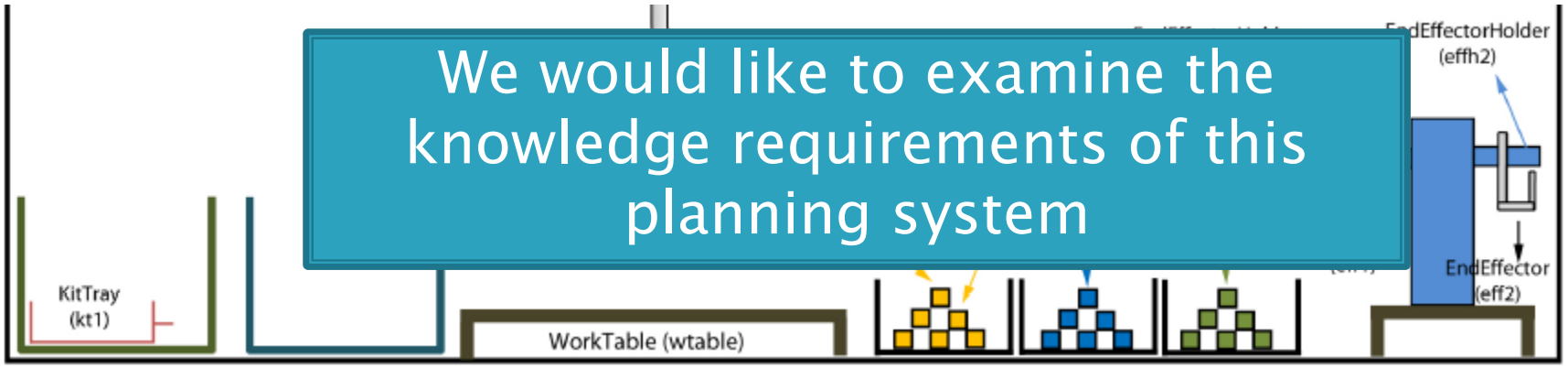
Kit Building

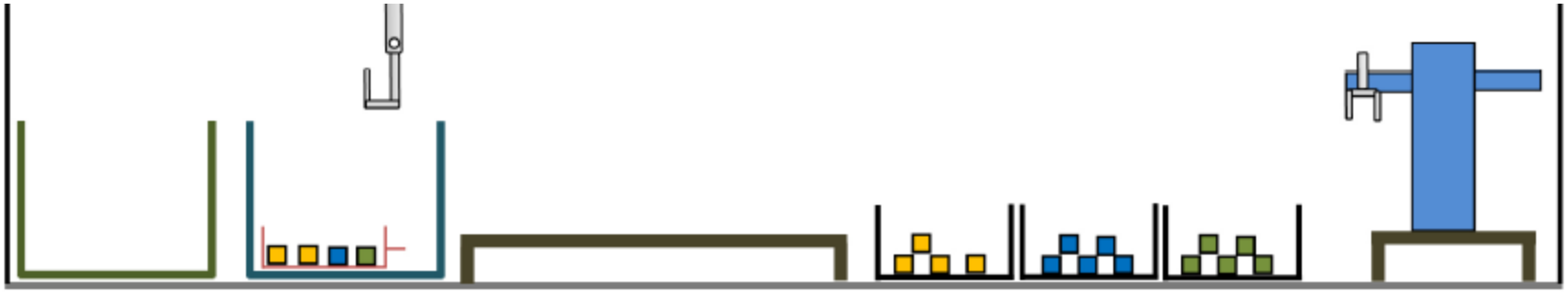
- Process in which individually separate but related items are grouped, packaged, and supplied together as one unit
- Utilized in many manufacturing assembly lines
- Robotic solution requires:
  - Flexibility –Many parts with varying characteristics, part-to-part inconsistences
  - Agility – Changes in part supply locations, lack of fixturing
  - Rapid re-tasking – Ability to quickly build new varieties of kits

Planning Problem

$$\mathcal{P} = (\Sigma, s_0, g)$$

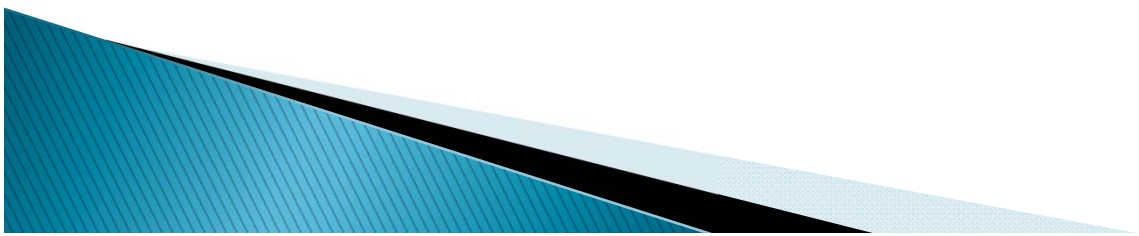
- Σ – State transition system (possible actions)
- S_0 – Initial conditions (objects and relationships)
- g – Goal conditions





Knowledge Foundation

- Capture a domain expert's knowledge
  - Actions
    - What actions and attributes are relevant
    - Necessary conditions for an action to occur
    - Likely results of the action
  - Objects
    - What objects and attributes are relevant
    - Taxonomy of objects
    - Relationships between objects



States and Actions

- State-Variable Representation[1] (SVR) used to formally define environment's state and actions
- Each state is represented by a tuple of values of *n* state variables $\{x_1,\ldots, x_n\}$
- Action is represented by a partial function that maps this tuple into some other tuple of values of the *n* state variables

[1]Dana Nau, Malik Ghallab, and Paolo Traverso. 2004. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc.

Compose SVR in Planning Domain Definition Language (PDDL)

- SVR is not a standard interchange language
  - Can use the PDDL to encode the SVR
- Entire kitting domain composed of 8 high-level operators: *take-kit-tray*,

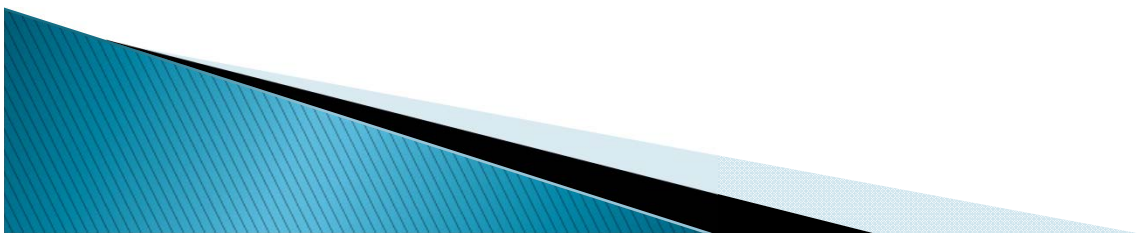
take-kt$(r,kt,lbwekt,eff,wtable)$: The *Robot* r equipped with the *EndEffector* $eeff$ picks up the *KitTray* kt from the *LargeBoxWithEmptyKitTrays* $lbwekt$.

precond	*effects*
rhold-empty(r),	¬rhold-empty(r),
lbwekt-non-empty$(lbwekt)$,	ktlocation(kt,r),
r-with-eff(r,eff),	rhold(r,kt),
ktlocation$(kt,lbwekt)$,	¬ktlocation$(kt,lbwekt)$
efflocation(eff,r),	
worktable-empty$(wtable)$,	
efftype(eff,kt)	

SVR – PDDL Limitations

- Lacks a taxonomy of objects
- No representation of relationships or constraints between objects
- No ability to reason or infer information

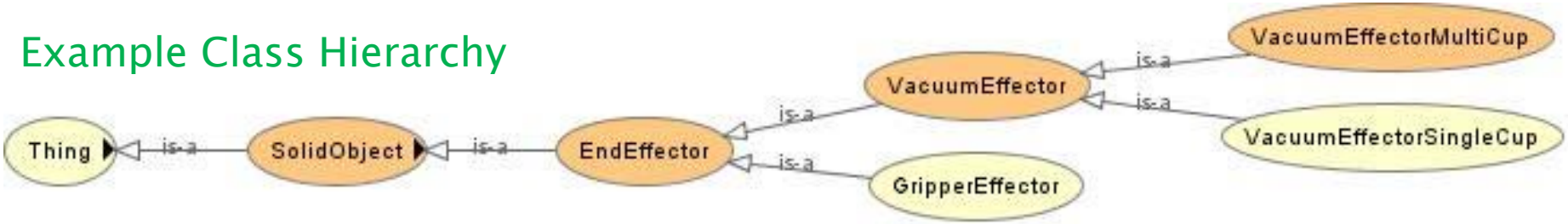
- An ontology provides for all of the above through entities, data properties, and object properties



Ontology

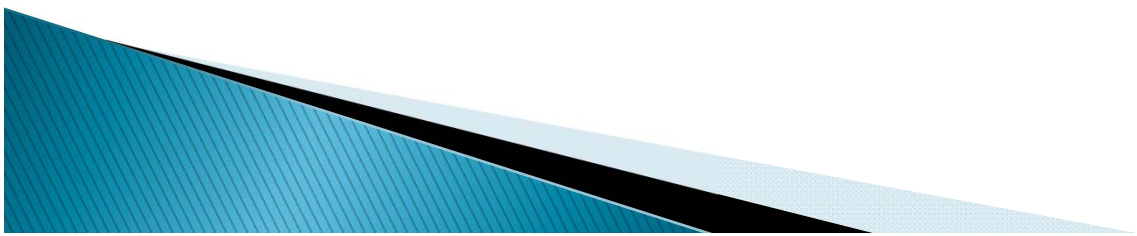
- IEEE Robotics and Automation Society's Ontologies for Robotics and Automation Working Group has chosen the Web Ontology Language (OWL) as the representation language

Example Class Hierarchy

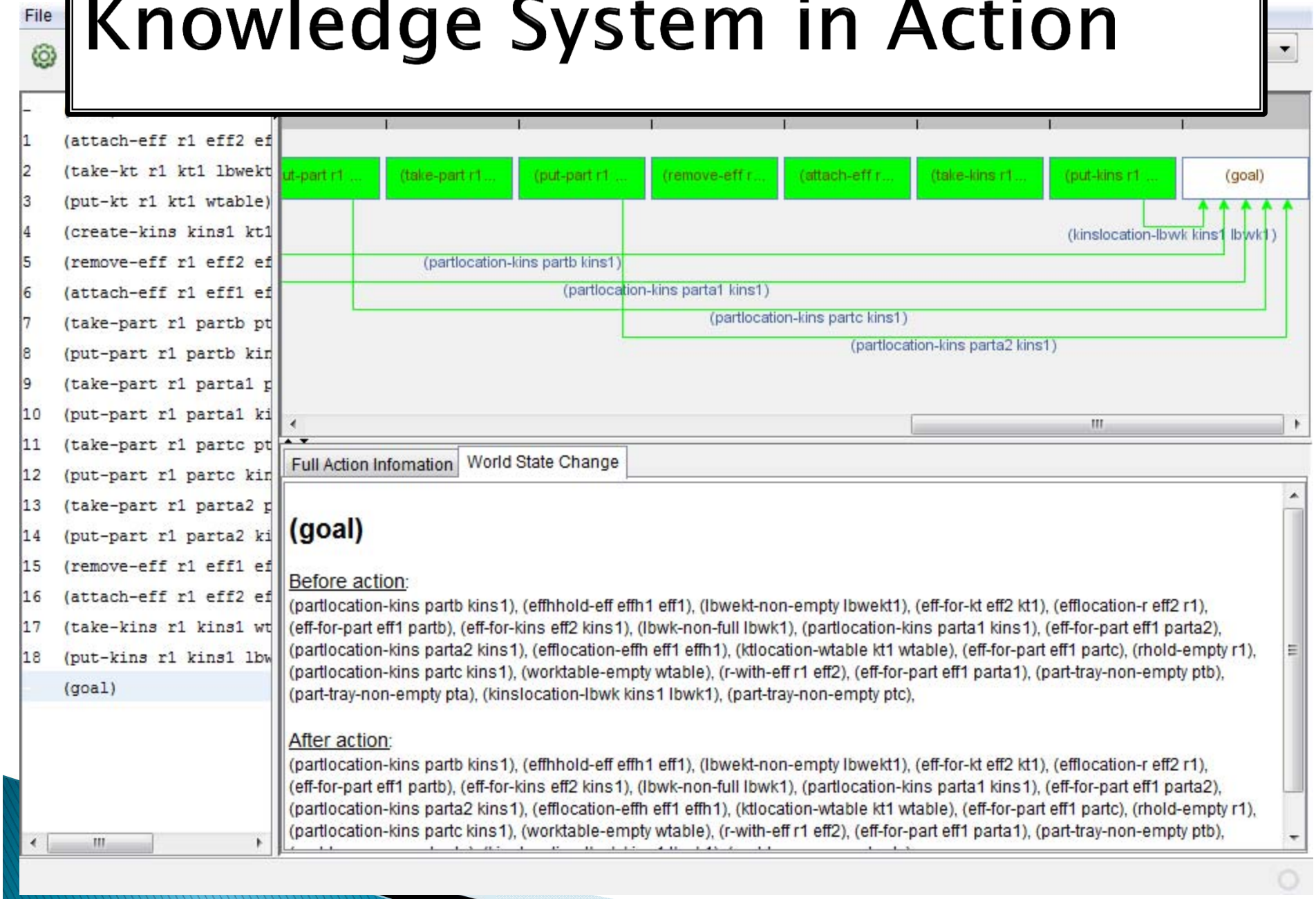


Limitations of OWL

- Set of static files with no real-time information
  - Can auto-generate MySQL database from OWL instance file and maintain database with sensor processing
  - Can re-generate OWL instance file to allow for continued reasoning
- No way to encode preconditions and effects of actions
  - OWL Services (OWL-S) and the Semantic Web Rule Language (SWRL) are extensions that provide for actions with preconditions and effects



Knowledge System in Action



PDDL Plan Instance File

(attach-eff r1 eff2 effh2)

Before action:
(ktlocation-lbwekt kt1 lbwekt1), (effhhold-eff effh1 eff1), ~~(effhhold-eff effh2 eff2)~~, (lbwekt-non-empty lbwekt1), (eff-for-kt eff2 kt1), (eff-for-part eff1 partb), (eff-for-kins eff2 kins1), (lbwk-non-full lbwk1), ~~(efflocation-effh eff2 effh2)~~, (eff-for-part eff1 parta2), (efflocation-effh eff1 effh1), (partlocation-pt partb ptb), (eff-for-part eff1 partc), ~~(r-no-eff r1)~~, (worktable-empty wtable), (partlocation-pt partc ptc), (partlocation-pt parta2 pta), (partlocation-pt parta1 pta), (eff-for-part eff1 parta1), (part-tray-non-empty ptb), (part-tray-non-empty pta), (part-tray-non-empty ptc),

After action:
(ktlocation-lbwekt kt1 lbwekt1), (effhhold-eff effh1 eff1), (lbwekt-non-empty lbwekt1), (eff-for-kt eff2 kt1), (efflocation-r eff2 r1), (eff-for-part eff1 partb), (eff-for-kins eff2 kins1), (lbwk-non-full lbwk1), (eff-for-part eff1 parta2), (efflocation-effh eff1 effh1), (partlocation-pt partb ptb), (eff-for-part eff1 partc), (rhold-empty r1), (worktable-empty wtable), (r-with-eff r1 eff2), (partlocation-pt partc ptc), (partlocation-pt parta2 pta), (partlocation-pt parta1 pta), (eff-for-part eff1 parta1), (part-tray-non-empty ptb), (part-tray-non-empty pta), (part-tray-non-empty ptc),

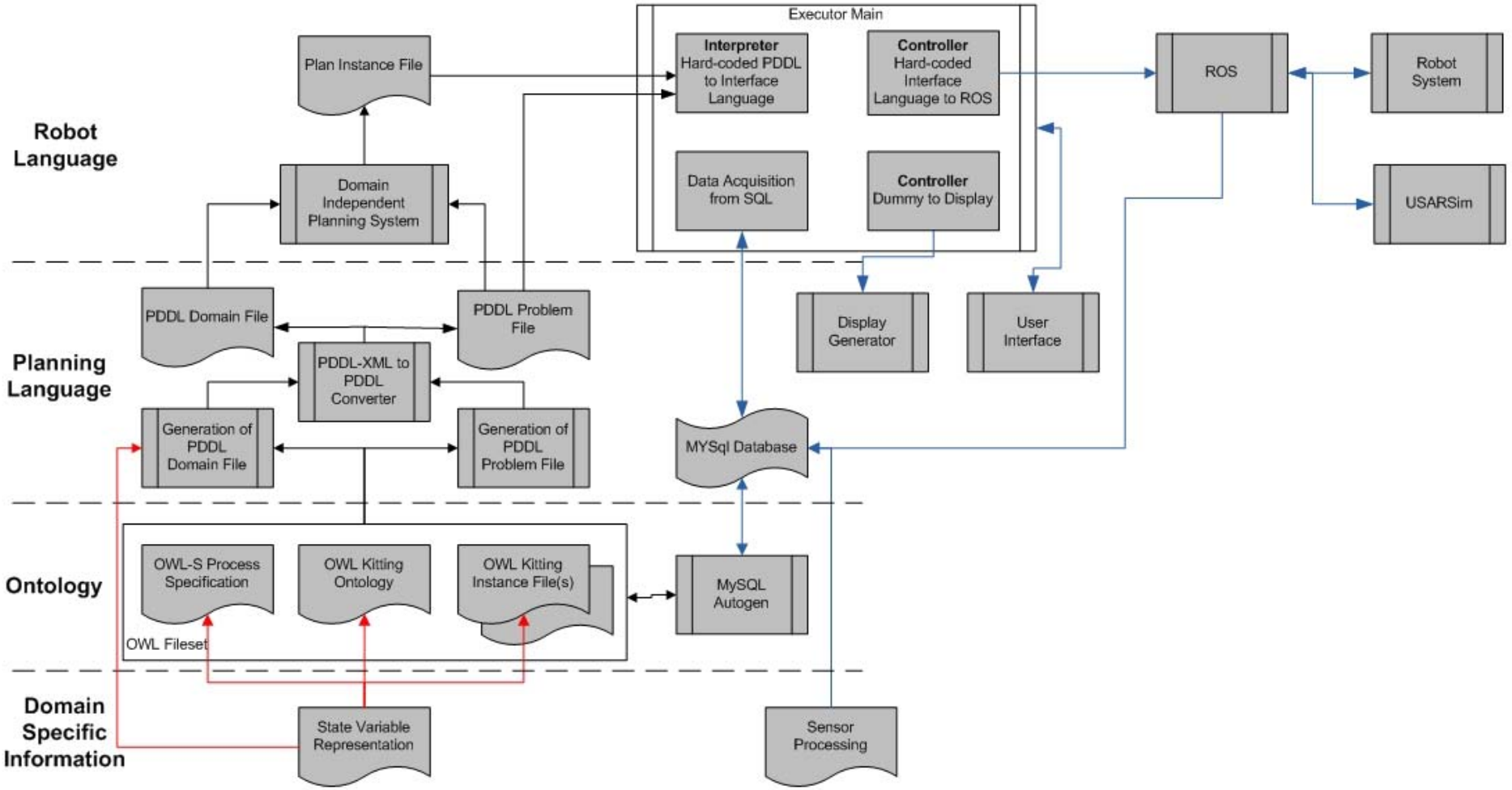
- Intentionally incomplete information
  - Command provides task-level information; most robots will not know how to “attach-eff”
  - Specific information needed to accomplish task is missing
  - This intentional lack of information provides for decoupling of task knowledge and environment knowledge
- Executor module populates the details
  - Combines task knowledge from PDDL with environment knowledge from MySQL
  - Utilizes “Canonical Robot Command Language” as an Interlingua

Example: attach-eff r1 eff2 effh2

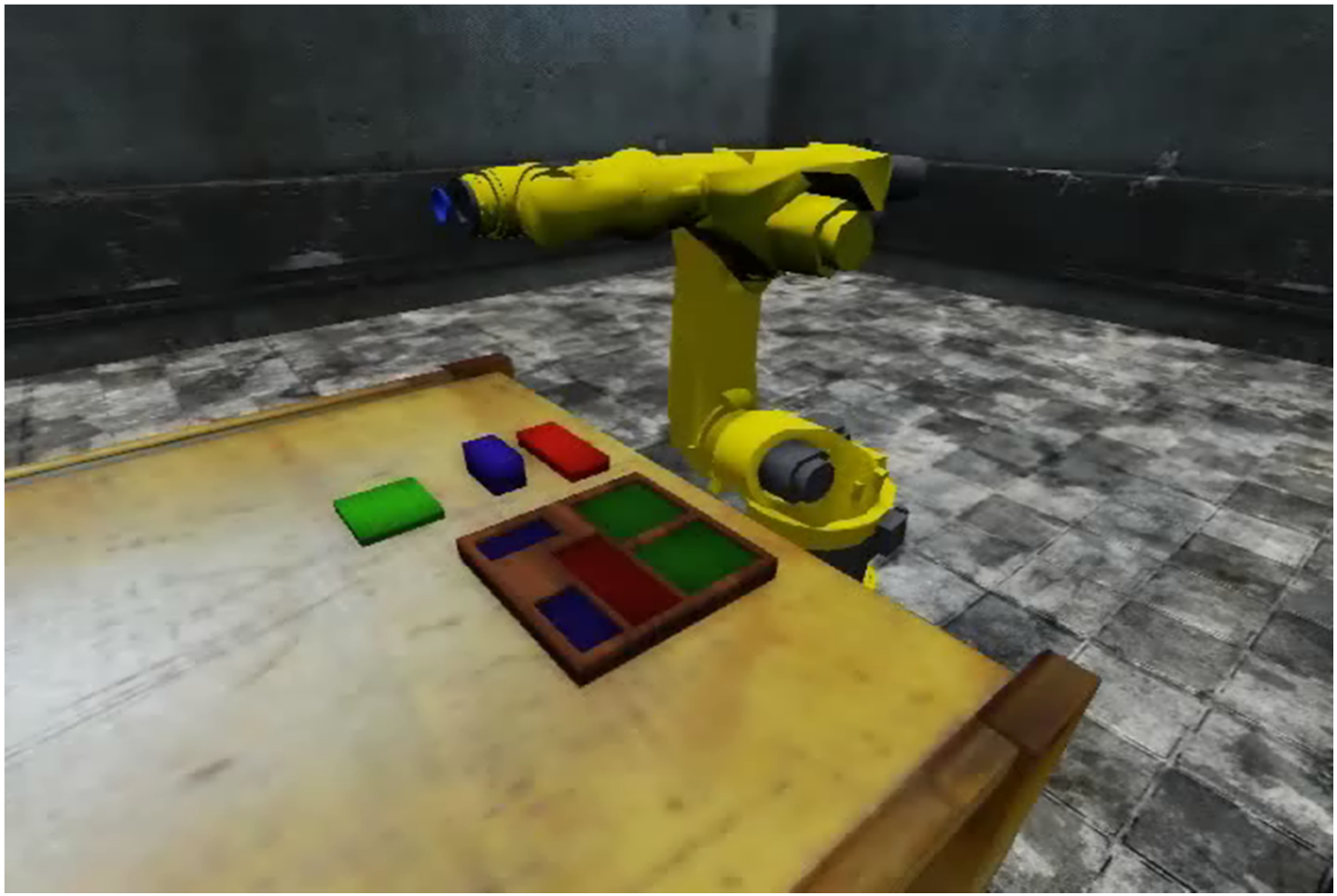
Subcommand	PDDL	MySQL	Data Update
Move	R1, effh2	Actual locations, offsets	---
Actuate	R1		R1 holds eff2 Effh2 empty
Move	R1	Offsets	---

- Must have mapping between PDDL actions and required robot sub-actions
- Must update knowledge base (MySQL)
- MySQL knowledge base contains all data from ontology needed by planner

Complete Knowledge Architecture



Operational System



Video is 2x real time

Future Work

- Current work assumes perfect actions; need techniques to gracefully cope with errors
- Migrate techniques onto real hardware with real image processing
- Extend work to apply to general assembly operations