plasp: A Prototype for PDDL-Based Planning in ASP

Martin Gebser Roland Kaminski Murat Knecht
Torsten Schaub

University of Potsdam

Outline

1. Motivation

2. System

3. Summary

Outline

1 Motivation

2 System

3 Summary

Automated Planning

Planning as Satisfiability

Input First-order description of a planning problem

☞ Planning Domain Definition Language (PDDL)

Approach Plans by (propositional) models

1. Knowledge representation
2. Incremental instantiation
3. Model generation (search)

Advantage Utilization of powerful general-purpose techniques

- Boolean Satisfiability (SAT)
  3. (Incremental) search
- Answer Set Programming (ASP)
  1. Knowledge representation
  2. (Incremental) instantiation
  3. (Incremental) search

Automated Planning

Planning as Satisfiability

Input First-order description of a planning problem

☞ Planning Domain Definition Language (PDDL)

Approach Plans by (propositional) models

1. Knowledge representation
2. Incremental instantiation
3. Model generation (search)

Advantage Utilization of powerful general-purpose techniques

- Boolean Satisfiability (SAT)
  3. (Incremental) search
- Answer Set Programming (ASP)
  1. Knowledge representation
  2. (Incremental) instantiation
  3. (Incremental) search

Automated Planning

Planning as Satisfiability

Input First-order description of a planning problem

☞ Planning Domain Definition Language (PDDL)

Approach Plans by (propositional) models

1. Knowledge representation
2. Incremental instantiation
3. Model generation (search)

Advantage Utilization of powerful general-purpose techniques

- Boolean Satisfiability (SAT)
  3. (Incremental) search
- Answer Set Programming (ASP)
  1. Knowledge representation
  2. (Incremental) instantiation
  3. (Incremental) search

Automated Planning

Planning as Satisfiability

Input: First-order description of a planning problem

☞ Planning Domain Definition Language (PDDL)

Approach: Plans by (propositional) models

1. Knowledge representation
2. Incremental instantiation
3. Model generation (search)

Advantage: Utilization of powerful general-purpose techniques

- Boolean Satisfiability (SAT)
  3. (Incremental) search
- Answer Set Programming (ASP)
  1. Knowledge representation
  2. (Incremental) instantiation
  3. (Incremental) search

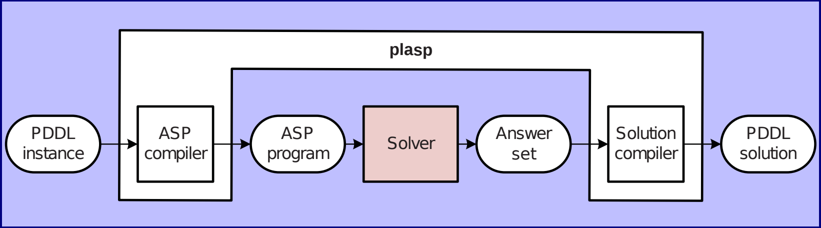
Outline

1 Motivation

2 System

3 Summary

Overview of *plasp*



Components

ASP compiler generates ASP program from PDDL input files

☞ several ASP encodings selectable (via options)

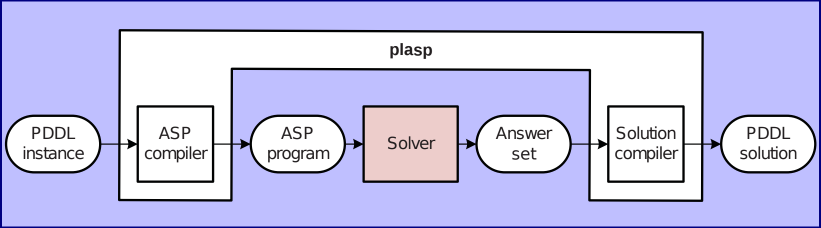
Solver incremental ASP system *iClingo* (version 2.0.5)

☞ successive grounding and solving till answer set

Solution compiler generates PDDL plan from answer set

☞ including parallel plan sequentialization

Overview of *plasp*



Components

ASP compiler generates ASP program from PDDL input files

☞ several ASP encodings selectable (via options)

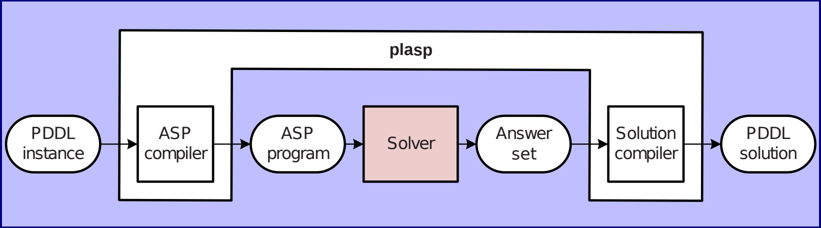
Solver incremental ASP system *iClingo* (version 2.0.5)

☞ successive grounding and solving till answer set

Solution compiler generates PDDL plan from answer set

☞ including parallel plan sequentialization

Overview of *plasp*



Components

ASP compiler generates ASP program from PDDL input files

☞ several ASP encodings selectable (via options)

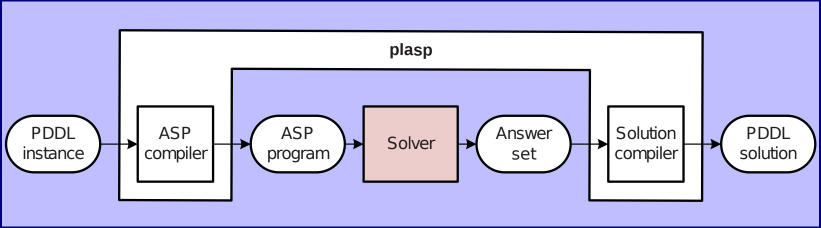
Solver incremental ASP system *iClingo* (version 2.0.5)

☞ successive grounding and solving till answer set

Solution compiler generates PDDL plan from answer set

☞ including parallel plan sequentialization

Overview of *plasp*



Components

ASP compiler generates ASP program from PDDL input files

☞ several ASP encodings selectable (via options)

Solver incremental ASP system *iClingo* (version 2.0.5)

☞ successive grounding and solving till answer set

Solution compiler generates PDDL plan from answer set

☞ including parallel plan sequentialization

Basic Meta-Encoding

ASP compilation

- PDDL (STRIPS fragment) instance mapped to facts
- Predicates: *init*, *goal*, *action*, *demands*, *adds*, *deletes*

(Incremental) ASP encoding

(1) $\#base.$

(2) $holds(F, 0) \leftarrow init(F).$

(3) $\#cumulative\ t.$

(4) $1\ \{apply(A, t) : action(A)\}\ 1.$

(5) $\leftarrow apply(A, t), demands(A, F, true), not\ holds(F, t{-}1).$

(6) $\leftarrow apply(A, t), demands(A, F, false), holds(F, t{-}1).$

(7) $holds(F, t) \leftarrow apply(A, t), adds(A, F).$

(8) $del(F, t) \leftarrow apply(A, t), deletes(A, F).$

(9) $holds(F, t) \leftarrow holds(F, t{-}1), not\ del(F, t).$

(10) $\#volatile\ t.$

(11) $\leftarrow goal(F, true), not\ holds(F, t).$

(12) $\leftarrow goal(F, false), holds(F, t).$

Basic Meta-Encoding

ASP compilation

- PDDL (STRIPS fragment) instance mapped to facts
- Predicates: *init*, *goal*, *action*, *demands*, *adds*, *deletes*

(Incremental) ASP encoding

(1) $\#base.$

(2) $holds(F,0) \leftarrow init(F).$

(3) $\#cumulative\ t.$

(4) $1\ \{apply(A,t) : action(A)\}\ 1.$

(5) $\leftarrow apply(A,t), demands(A,F,true), not\ holds(F,t{-}1).$

(6) $\leftarrow apply(A,t), demands(A,F,false), holds(F,t{-}1).$

(7) $holds(F,t) \leftarrow apply(A,t), adds(A,F).$

(8) $del(F,t) \leftarrow apply(A,t), deletes(A,F).$

(9) $holds(F,t) \leftarrow holds(F,t{-}1), not\ del(F,t).$

(10) $\#volatile\ t.$

(11) $\leftarrow goal(F,true), not\ holds(F,t).$

(12) $\leftarrow goal(F,false), holds(F,t).$

Basic Meta-Encoding

ASP compilation

- PDDL (STRIPS fragment) instance mapped to facts
- Predicates: *init*, *goal*, *action*, *demands*, *adds*, *deletes*

(Incremental) ASP encoding

(1) $\#\mathit{base}.$

(2) $\mathit{holds}(F,0) \leftarrow \mathit{init}(F).$

(3) $\#\mathit{cumulative}\ t.$

(4) $1\ \{\mathit{apply}(A,t) : \mathit{action}(A)\}\ 1.$

(5) $\leftarrow \mathit{apply}(A,t), \mathit{demands}(A,F,\mathit{true}), \mathit{not}\ \mathit{holds}(F,t-1).$

(6) $\leftarrow \mathit{apply}(A,t), \mathit{demands}(A,F,\mathit{false}), \mathit{holds}(F,t-1).$

(7) $\mathit{holds}(F,t) \leftarrow \mathit{apply}(A,t), \mathit{adds}(A,F).$

(8) $\mathit{del}(F,t) \leftarrow \mathit{apply}(A,t), \mathit{deletes}(A,F).$

(9) $\mathit{holds}(F,t) \leftarrow \mathit{holds}(F,t-1), \mathit{not}\ \mathit{del}(F,t).$

(10) $\#\mathit{volatile}\ t.$

(11) $\leftarrow \mathit{goal}(F,\mathit{true}), \mathit{not}\ \mathit{holds}(F,t).$

(12) $\leftarrow \mathit{goal}(F,\mathit{false}), \mathit{holds}(F,t).$

Basic Meta-Encoding

ASP compilation

- PDDL (STRIPS fragment) instance mapped to facts
- Predicates: *init*, *goal*, *action*, *demands*, *adds*, *deletes*

(Incremental) ASP encoding

(1) $\#\mathit{base}.$

(2) $\mathit{holds}(F,0) \leftarrow \mathit{init}(F).$

(3) $\#\mathit{cumulative}\ t.$

(4) $1\ \{\mathit{apply}(A,t) : \mathit{action}(A)\}\ 1.$

(5) $\leftarrow \mathit{apply}(A,t), \mathit{demands}(A,F,\mathit{true}), \mathit{not}\ \mathit{holds}(F,t{-}1).$

(6) $\leftarrow \mathit{apply}(A,t), \mathit{demands}(A,F,\mathit{false}), \mathit{holds}(F,t{-}1).$

(7) $\mathit{holds}(F,t) \leftarrow \mathit{apply}(A,t), \mathit{adds}(A,F).$

(8) $\mathit{del}(F,t) \leftarrow \mathit{apply}(A,t), \mathit{deletes}(A,F).$

(9) $\mathit{holds}(F,t) \leftarrow \mathit{holds}(F,t{-}1), \mathit{not}\ \mathit{del}(F,t).$

(10) $\#\mathit{volatile}\ t.$

(11) $\leftarrow \mathit{goal}(F,\mathit{true}), \mathit{not}\ \mathit{holds}(F,t).$

(12) $\leftarrow \mathit{goal}(F,\mathit{false}), \mathit{holds}(F,t).$

(More) ASP Encodings

Meta-Encodings

basic	sequential planning
concur	parallel planning
	☞ concurrent actions under ∀-step semantics
split	operator splitting
	☞ fixed arity of function terms for actions

Compilations

expand	on-the-fly instantiation of actions
	☞ instantiation based on preconditions' feasibility
concur+expand	parallel planning w.r.t. instantiated actions
split+expand	operator splitting w.r.t. instantiated actions

(More) ASP Encodings

Meta-Encodings

basic	sequential planning
concur	parallel planning
☞	concurrent actions under ∀-step semantics
split	operator splitting
☞	fixed arity of function terms for actions

Compilations

expand	on-the-fly instantiation of actions
☞	instantiation based on preconditions' feasibility
concur+expand	parallel planning w.r.t. instantiated actions
split+expand	operator splitting w.r.t. instantiated actions

Experiments

Benchmark	basic	concur	split	expand	concur+expand	split+expand	SATPlan	SGPlan
Blocks-4-0	0.16	0.21	0.43	0.21	0.22	0.20	0.34	0.10
Blocks-6-0	0.30	0.63	0.93	0.44	0.56	1.40	0.27	0.04
Blocks-8-0	1.58	6.53	12.78	98.60	317.53	47.57	1.24	0.09
Elevator-3-0	0.27	0.56	0.92	0.30	0.46	0.89	0.10	0.02
Elevator-4-0	11.72	264.11	20.30	14.69	324.18	28.88	0.30	0.02
Elevator-5-0	—	—	320.58	—	—	467.98	0.61	0.04
FreeCell-2-1	93.42	mem	64.28	60.52	51.33	56.94	2.44	0.12
FreeCell-3-1	—	mem	—	—	175.03	—	10.44	0.14
Logistics-4-0	7.85	0.38	79.15	8.81	0.39	70.56	0.34	0.05
Logistics-7-0	—	0.99	—	—	0.61	—	0.31	0.04
Logistics-9-0	—	0.89	—	—	0.57	—	0.27	0.04
Satellite-1	0.23	0.87	0.23	0.29	0.74	0.26	0.10	0.03
Satellite-2	4.56	638.08	2.19	5.43	448.60	2.69	0.41	0.03
Satellite-3	8.76	3.52	4.00	7.54	3.29	3.70	0.21	0.04
Schedule-2-0	mem	mem	mem	1.03	3.37	mem	mem	mem
Schedule-3-0	mem	mem	mem	1.63	12.89	mem	mem	mem

basic	*plasp*: sequential planning
concur	*plasp*: parallel planning
split	*plasp*: operator splitting
expand	*plasp*: on-the-fly instantiation of actions
concur+expand	*plasp*: parallel planning w.r.t. instantiated actions
split+expand	*plasp*: operator splitting w.r.t. instantiated actions
SATPlan	iterative compilation of parallel planning to SAT
SGPlan	heuristic (non-optimal) planning

Experiments

Benchmark	**basic**	concur	split	expand	concur+expand	split+expand	SATPlan	SGPlan
Blocks-4-0	**0.16**	0.21	0.43	0.21	0.22	0.20	0.34	0.10
Blocks-6-0	**0.30**	0.63	0.93	0.44	0.56	1.40	0.27	0.04
Blocks-8-0	**1.58**	6.53	12.78	98.60	317.53	47.57	1.24	0.09
Elevator-3-0	0.27	0.56	0.92	0.30	0.46	0.89	0.10	0.02
Elevator-4-0	11.72	264.11	20.30	14.69	324.18	28.88	0.30	0.02
Elevator-5-0	—	—	320.58	—	—	467.98	0.61	0.04
FreeCell-2-1	93.42	mem	64.28	60.52	51.33	56.94	2.44	0.12
FreeCell-3-1	—	mem	—	—	175.03	—	10.44	0.14
Logistics-4-0	7.85	0.38	79.15	8.81	0.39	70.56	0.34	0.05
Logistics-7-0	—	0.99	—	—	0.61	—	0.31	0.04
Logistics-9-0	—	0.89	—	—	0.57	—	0.27	0.04
Satellite-1	0.23	0.87	0.23	0.29	0.74	0.26	0.10	0.03
Satellite-2	4.56	638.08	2.19	5.43	448.60	2.69	0.41	0.03
Satellite-3	8.76	3.52	4.00	7.54	3.29	3.70	0.21	0.04
Schedule-2-0	mem	mem	mem	1.03	3.37	mem	mem	mem
Schedule-3-0	mem	mem	mem	1.63	12.89	mem	mem	mem

basic	*plasp*: sequential planning
concur	*plasp*: parallel planning
split	*plasp*: operator splitting
expand	*plasp*: on-the-fly instantiation of actions
concur+expand	*plasp*: parallel planning w.r.t. instantiated actions
split+expand	*plasp*: operator splitting w.r.t. instantiated actions
SATPlan	iterative compilation of parallel planning to SAT
SGPlan	heuristic (non-optimal) planning

Experiments

Benchmark	basic	concur	**split**	expand	concur+expand	split+expand	SATPlan	SGPlan
Blocks-4-0	0.16	0.21	0.43	0.21	0.22	0.20	0.34	0.10
Blocks-6-0	0.30	0.63	0.93	0.44	0.56	1.40	0.27	0.04
Blocks-8-0	1.58	6.53	12.78	98.60	317.53	47.57	1.24	0.09
Elevator-3-0	0.27	0.56	**0.92**	0.30	0.46	0.89	0.10	0.02
Elevator-4-0	11.72	264.11	**20.30**	14.69	324.18	28.88	0.30	0.02
Elevator-5-0	—	—	**320.58**	—	—	467.98	0.61	0.04
FreeCell-2-1	93.42	mem	64.28	60.52	51.33	56.94	2.44	0.12
FreeCell-3-1	—	mem	—	—	175.03	—	10.44	0.14
Logistics-4-0	7.85	0.38	79.15	8.81	0.39	70.56	0.34	0.05
Logistics-7-0	—	0.99	—	—	0.61	—	0.31	0.04
Logistics-9-0	—	0.89	—	—	0.57	—	0.27	0.04
Satellite-1	0.23	0.87	0.23	0.29	0.74	0.26	0.10	0.03
Satellite-2	4.56	638.08	2.19	5.43	448.60	2.69	0.41	0.03
Satellite-3	8.76	3.52	4.00	7.54	3.29	3.70	0.21	0.04
Schedule-2-0	mem	mem	mem	1.03	3.37	mem	mem	mem
Schedule-3-0	mem	mem	mem	1.63	12.89	mem	mem	mem

basic	*plasp*: sequential planning
concur	*plasp*: parallel planning
split	*plasp*: operator splitting
expand	*plasp*: on-the-fly instantiation of actions
concur+expand	*plasp*: parallel planning w.r.t. instantiated actions
split+expand	*plasp*: operator splitting w.r.t. instantiated actions
SATPlan	iterative compilation of parallel planning to SAT
SGPlan	heuristic (non-optimal) planning

Experiments

Benchmark	basic	concur	split	expand	**concur+expand**	split+expand	SATPlan	SGPlan
Blocks-4-0	0.16	0.21	0.43	0.21	0.22	0.20	0.34	0.10
Blocks-6-0	0.30	0.63	0.93	0.44	0.56	1.40	0.27	0.04
Blocks-8-0	1.58	6.53	12.78	98.60	317.53	47.57	1.24	0.09
Elevator-3-0	0.27	0.56	0.92	0.30	0.46	0.89	0.10	0.02
Elevator-4-0	11.72	264.11	20.30	14.69	324.18	28.88	0.30	0.02
Elevator-5-0	—	—	320.58	—	—	467.98	0.61	0.04
FreeCell-2-1	93.42	mem	64.28	60.52	**51.33**	56.94	2.44	0.12
FreeCell-3-1	—	mem	—	—	**175.03**	—	10.44	0.14
Logistics-4-0	7.85	0.38	79.15	8.81	0.39	70.56	0.34	0.05
Logistics-7-0	—	0.99	—	—	0.61	—	0.31	0.04
Logistics-9-0	—	0.89	—	—	0.57	—	0.27	0.04
Satellite-1	0.23	0.87	0.23	0.29	0.74	0.26	0.10	0.03
Satellite-2	4.56	638.08	2.19	5.43	448.60	2.69	0.41	0.03
Satellite-3	8.76	3.52	4.00	7.54	3.29	3.70	0.21	0.04
Schedule-2-0	mem	mem	mem	1.03	3.37	mem	mem	mem
Schedule-3-0	mem	mem	mem	1.63	12.89	mem	mem	mem

basic	*plasp*: sequential planning
concur	*plasp*: parallel planning
split	*plasp*: operator splitting
expand	*plasp*: on-the-fly instantiation of actions
concur+expand	*plasp*: parallel planning w.r.t. instantiated actions
split+expand	*plasp*: operator splitting w.r.t. instantiated actions
SATPlan	iterative compilation of parallel planning to SAT
SGPlan	heuristic (non-optimal) planning

Experiments

Benchmark	basic	**concur**	split	expand	**concur+expand**	split+expand	SATPlan	SGPlan
Blocks-4-0	0.16	0.21	0.43	0.21	0.22	0.20	0.34	0.10
Blocks-6-0	0.30	0.63	0.93	0.44	0.56	1.40	0.27	0.04
Blocks-8-0	1.58	6.53	12.78	98.60	317.53	47.57	1.24	0.09
Elevator-3-0	0.27	0.56	0.92	0.30	0.46	0.89	0.10	0.02
Elevator-4-0	11.72	264.11	20.30	14.69	324.18	28.88	0.30	0.02
Elevator-5-0	—	—	320.58	—	—	467.98	0.61	0.04
FreeCell-2-1	93.42	mem	64.28	60.52	51.33	56.94	2.44	0.12
FreeCell-3-1	—	mem	—	—	175.03	—	10.44	0.14
Logistics-4-0	7.85	**0.38**	79.15	8.81	**0.39**	70.56	0.34	0.05
Logistics-7-0	—	**0.99**	—	—	**0.61**	—	0.31	0.04
Logistics-9-0	—	**0.89**	—	—	**0.57**	—	0.27	0.04
Satellite-1	0.23	0.87	0.23	0.29	0.74	0.26	0.10	0.03
Satellite-2	4.56	638.08	2.19	5.43	448.60	2.69	0.41	0.03
Satellite-3	8.76	3.52	4.00	7.54	3.29	3.70	0.21	0.04
Schedule-2-0	mem	mem	mem	1.03	3.37	mem	mem	mem
Schedule-3-0	mem	mem	mem	1.63	12.89	mem	mem	mem

basic	*plasp*: sequential planning
concur	*plasp*: parallel planning
split	*plasp*: operator splitting
expand	*plasp*: on-the-fly instantiation of actions
concur+expand	*plasp*: parallel planning w.r.t. instantiated actions
split+expand	*plasp*: operator splitting w.r.t. instantiated actions
SATPlan	iterative compilation of parallel planning to SAT
SGPlan	heuristic (non-optimal) planning

Experiments

Benchmark	basic	concur	**split**	expand	concur+expand	**split+expand**	SATPlan	SGPlan
Blocks-4-0	0.16	0.21	0.43	0.21	0.22	0.20	0.34	0.10
Blocks-6-0	0.30	0.63	0.93	0.44	0.56	1.40	0.27	0.04
Blocks-8-0	1.58	6.53	12.78	98.60	317.53	47.57	1.24	0.09
Elevator-3-0	0.27	0.56	0.92	0.30	0.46	0.89	0.10	0.02
Elevator-4-0	11.72	264.11	20.30	14.69	324.18	28.88	0.30	0.02
Elevator-5-0	—	—	320.58	—	—	467.98	0.61	0.04
FreeCell-2-1	93.42	mem	64.28	60.52	51.33	56.94	2.44	0.12
FreeCell-3-1	—	mem	—	—	175.03	—	10.44	0.14
Logistics-4-0	7.85	0.38	79.15	8.81	0.39	70.56	0.34	0.05
Logistics-7-0	—	0.99	—	—	0.61	—	0.31	0.04
Logistics-9-0	—	0.89	—	—	0.57	—	0.27	0.04
Satellite-1	0.23	0.87	**0.23**	0.29	0.74	**0.26**	0.10	0.03
Satellite-2	4.56	638.08	**2.19**	5.43	448.60	**2.69**	0.41	0.03
Satellite-3	8.76	3.52	**4.00**	7.54	3.29	**3.70**	0.21	0.04
Schedule-2-0	mem	mem	mem	1.03	3.37	mem	mem	mem
Schedule-3-0	mem	mem	mem	1.63	12.89	mem	mem	mem

basic *plasp*: sequential planning

concur *plasp*: parallel planning

split *plasp*: operator splitting

expand *plasp*: on-the-fly instantiation of actions

concur+expand *plasp*: parallel planning w.r.t. instantiated actions

split+expand *plasp*: operator splitting w.r.t. instantiated actions

SATPlan iterative compilation of parallel planning to SAT

SGPlan heuristic (non-optimal) planning

Experiments

Benchmark	basic	concur	split	**expand**	concur+expand	split+expand	SATPlan	SGPlan
Blocks-4-0	0.16	0.21	0.43	0.21	0.22	0.20	0.34	0.10
Blocks-6-0	0.30	0.63	0.93	0.44	0.56	1.40	0.27	0.04
Blocks-8-0	1.58	6.53	12.78	98.60	317.53	47.57	1.24	0.09
Elevator-3-0	0.27	0.56	0.92	0.30	0.46	0.89	0.10	0.02
Elevator-4-0	11.72	264.11	20.30	14.69	324.18	28.88	0.30	0.02
Elevator-5-0	—	—	320.58	—	—	467.98	0.61	0.04
FreeCell-2-1	93.42	mem	64.28	60.52	51.33	56.94	2.44	0.12
FreeCell-3-1	—	mem	—	—	175.03	—	10.44	0.14
Logistics-4-0	7.85	0.38	79.15	8.81	0.39	70.56	0.34	0.05
Logistics-7-0	—	0.99	—	—	0.61	—	0.31	0.04
Logistics-9-0	—	0.89	—	—	0.57	—	0.27	0.04
Satellite-1	0.23	0.87	0.23	0.29	0.74	0.26	0.10	0.03
Satellite-2	4.56	638.08	2.19	5.43	448.60	2.69	0.41	0.03
Satellite-3	8.76	3.52	4.00	7.54	3.29	3.70	0.21	0.04
Schedule-2-0	mem	mem	mem	**1.03**	3.37	mem	mem	mem
Schedule-3-0	mem	mem	mem	**1.63**	12.89	mem	mem	mem

basic	*plasp*: sequential planning
concur	*plasp*: parallel planning
split	*plasp*: operator splitting
expand	*plasp*: on-the-fly instantiation of actions
concur+expand	*plasp*: parallel planning w.r.t. instantiated actions
split+expand	*plasp*: operator splitting w.r.t. instantiated actions
SATPlan	iterative compilation of parallel planning to SAT
SGPlan	heuristic (non-optimal) planning

Outline

1 Motivation

2 System

3 Summary

Discussion

plasp

- prototype for PDDL-based planning in (incremental) ASP
  - ☞ various ASP (meta-)encodings
- `http://potassco.sourceforge.net`

Outlook

- optimization/extension of ASP encodings
  - ☞ on-the-fly instantiation inherently supported by *iClingo* 3.X.X
  - ☞ inclusion of further planning techniques (e.g., mutex analysis)
  - ☞ generalization beyond STRIPS fragment of PDDL
- exploitation of planning domains' characteristic properties
  - ☞ ASP-based induction proofs (applied to General Game Playing)
  - ☞ automatic encoding selection
- extension to further disciplines (e.g., Conformant Planning)

Discussion

plasp

- prototype for PDDL-based planning in (incremental) ASP
  - ☞ various ASP (meta-)encodings
- `http://potassco.sourceforge.net`

Outlook

- optimization/extension of ASP encodings
  - ☞ on-the-fly instantiation inherently supported by *iClingo* 3.X.X
  - ☞ inclusion of further planning techniques (e.g., mutex analysis)
  - ☞ generalization beyond STRIPS fragment of PDDL
- exploitation of planning domains' characteristic properties
  - ☞ ASP-based induction proofs (applied to General Game Playing)
  - ☞ automatic encoding selection
- extension to further disciplines (e.g., Conformant Planning)