Encoding Lifted Classical Planning in Propositional Logic

Daniel Höller[1] and Gregor Behnke[2,3]

[1]Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
[2]University of Freiburg, Freiburg, Germany
[3]University of Amsterdam, ILLC, The Netherlands
hoeller@cs.uni-saarland.de, g.behnke@uva.nl

Motivation

- Planning models are usually defined in lifted formalisms
- Most planning systems from the last decades need a variable-free, propositional model

→ Models are compiled (*grounded*) before planning

- Despite the used pruning techniques, this process might be prohibitory costly

→ Recently, several systems have been introduced that avoid grounding

- All approaches are based on search (mostly *heuristic* search)

→ We present a novel approach based on compilation to propositional logic

Approach

- We present a lifted planner based on propositional logic
- We avoid representing any (intermediate-)state
- Instead we capture the causal structure in the plan, inspired from Plan Space Planning
- We also keep actions lifted and encode equality constraints between parameters

Related Work

Other than earlier encodings, we encode a total ordering between actions

- This enables a quadratic encoding (instead of a cubic one)
- It allows for an optimization for satisficing planning

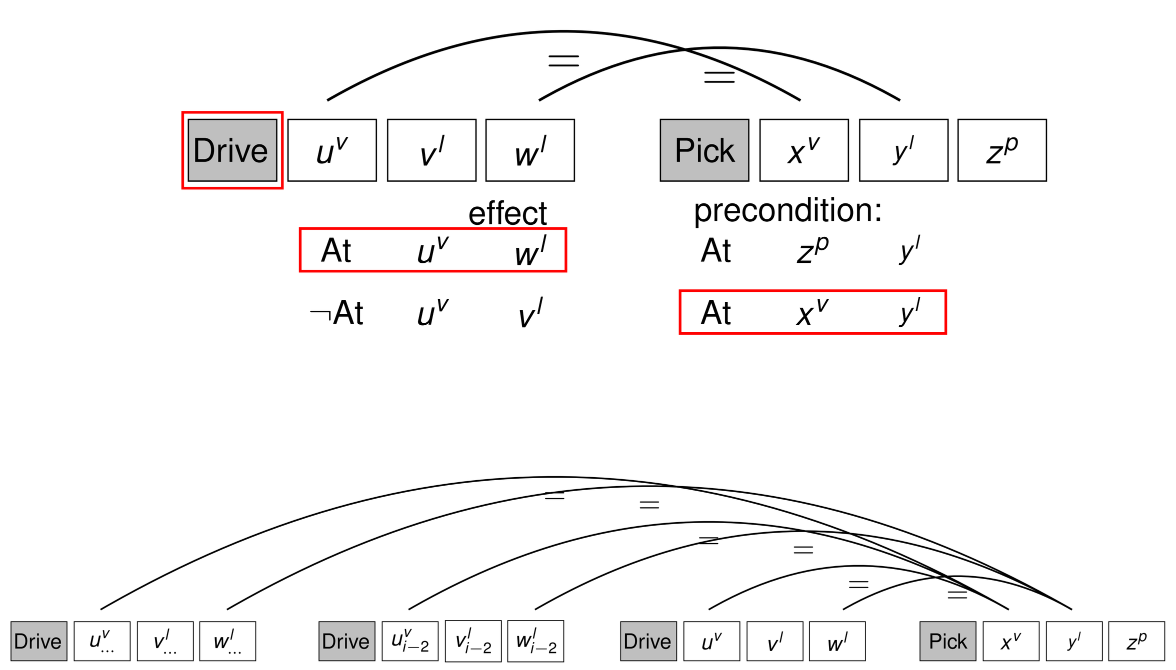

Figure 1: *Encoding – Basic idea (1).*

Encoding – Basic Idea

- For each action a (here: *Pickup*), each precondition (vehicle x^v at location y^l)...
- ...is achieved by s_0 or there is an action b (*Drive*) achieving the precondition...
- ...and no destroyer between achiever/consumer

$$(a_i = Pickup) \Rightarrow ((a_{i-1} = Drive) \wedge (u^v = x^v) \wedge (w^l = y^l)) \vee ((a_{i-2} = Drive) \wedge (u^v_{i-2} = x^v) \wedge (w^l_{i-2} = y^l)) \dots$$

Figure 2: *Encoding – Basic idea (2).*

Encoding – Observations (1)

- We do not need to represent intermediate states (only s_0)
- We do not need to constrain all variables, only those contained in the effect/precondition (*from* which position vehicle x^v drives to y^l is not relevant)
- We do not need to actually bind the variables, we need to assure equality

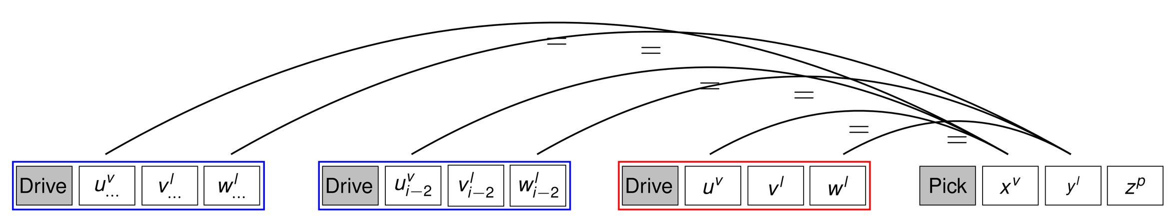

Figure 3: *Destroyer position.*

Encoding – Observations (2)

- Whether an action at pos. i destroys an achieved precondition does not depend on the exact position where it was achieved → encoding must not be repeated

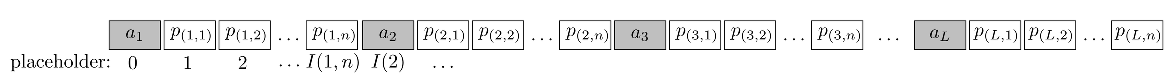

Figure 4: *Plan schema.*

Encoding – Propositional Logic

- We encode the resulting constraints in propositional logic
- (Main) variables describe:
  - Actions at each time step
  - Equality between parameters
  - That a parameter has a certain value
- We rearrange parameters to decrease the num. of constraints to encode
  - Parameter typing
  - (In-)equality between parameters
- We use a special encoding for s_0 to decrease its size

Implementation

- Implemented on top of the Powerlifted planner
- We tested also incremental solvers but did not find major improvements

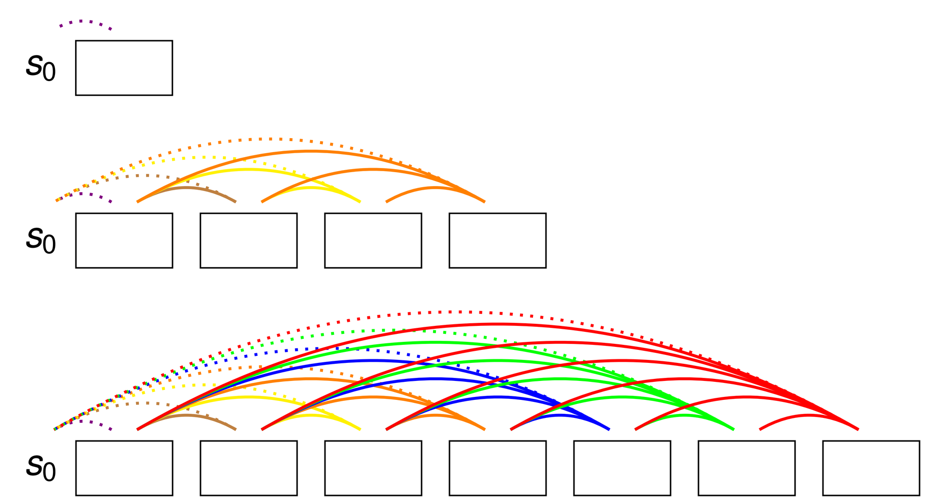

Figure 5: *Translation scheme for optimal planning.*

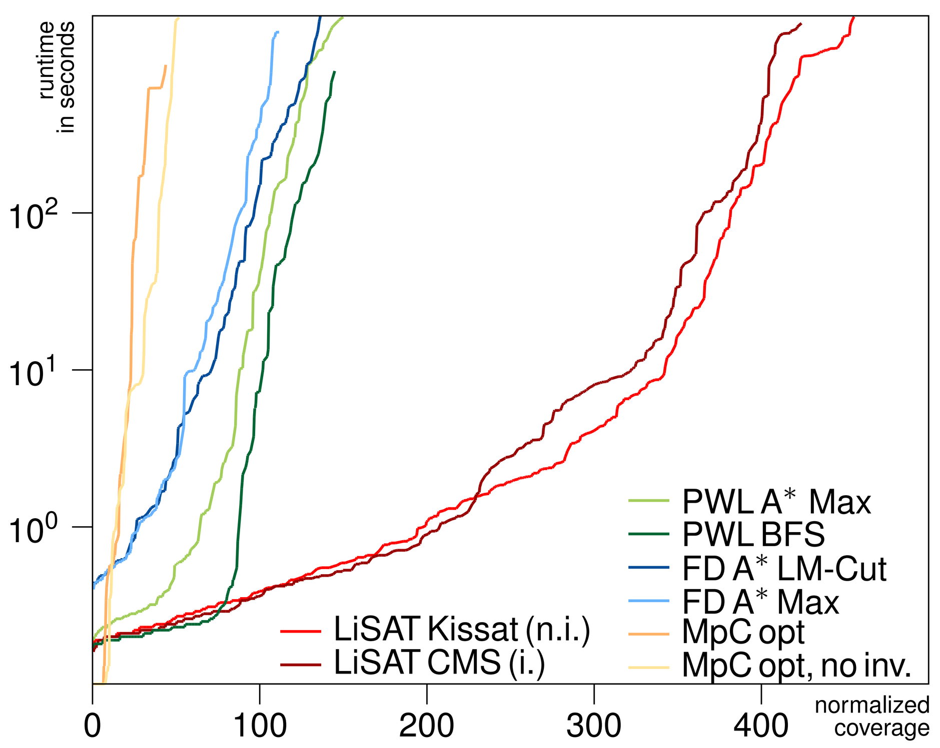

Figure 6: *Empirical evaluation – optimal planning.*

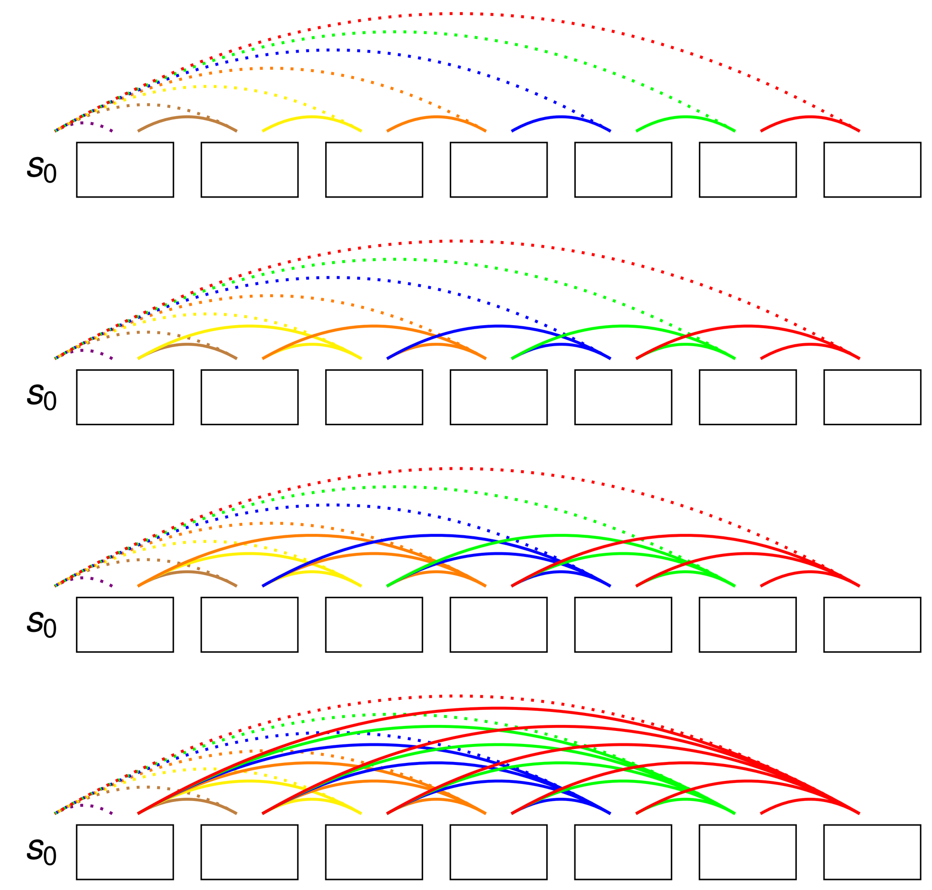

Figure 7: *Translation scheme for satisficing planning.*

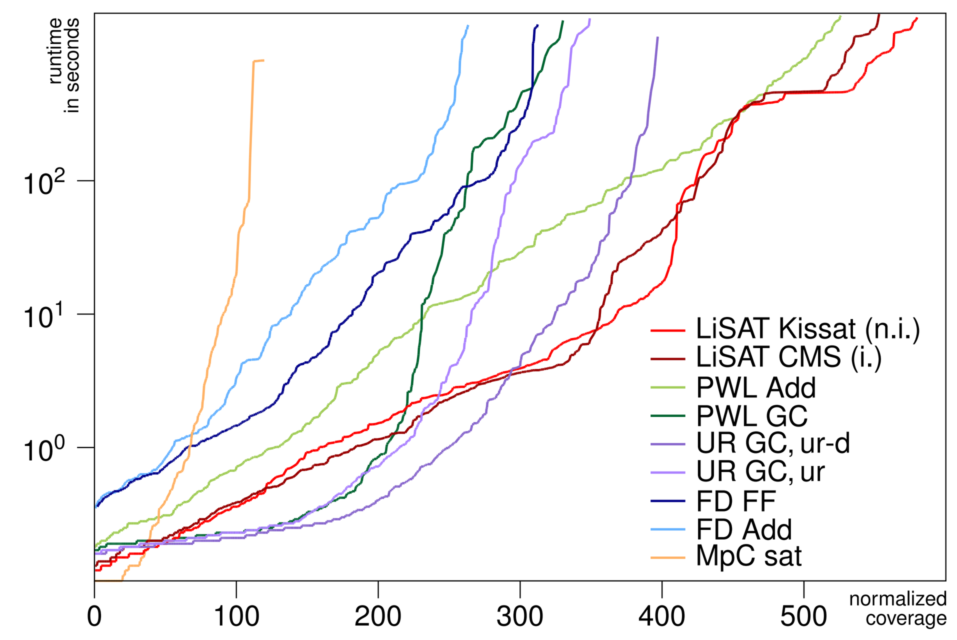

Figure 8: *Empirical evaluation – satisficing planning.*

Conclusion

- We introduced an encoding of lifted planning in prop. logic
- It is inspired by Plan Space planning
- It does not repr. interm. states, but encodes causal relations
- We present a translation scheme for optimal planning, incrementally increasing plan length and ...
- ...one for satisficing planning, further bounding the distance between a precondition of an action and the action fulfilling it
- We implemented our system in the Powerlifted planner and ...
- ...evaluated it on a benchmark set presented for lifted planning
- Our system outperforms the systems from the literature in optimal planning, and is also competitive in satisficing planning

UNIVERSITÄT DES SAARLANDES

UNI FREIBURG