Relaxing the Relaxed Exist-Step Parallel Planning Semantics

What is Planning?
- World states are described as values of state variables
- Actions change the state of the world by changing the values of state variables by their effects
- Actions also have preconditions and are applicable only when their preconditions hold in the given state

Objective: given a set a of actions, an initial world state and the description of a goal state find a valid sequence of actions that transforms the world from the initial state to a goal state

Using SAT solvers to solve planning problems
- Construct a formula $F_n$ such that it is satisfiable if and only if there is a plan of at most $k$ steps
- Solve $F_n$, $F_{n-1}$,... using a SAT solver until you reach a satisfiable formula $F_0$
- Extract a plan from the satisfying assignment of $F_0$

Example: delivering 2 packages to Las Vegas

State Variables and their domains:
- Truck location $T$, dom($T$) = {LA, SF, LV}
- Package locations $P$ and $Q$
  dom($P$) = dom($Q$) = {LA, SF, LV, Tr}

Initial State: $T$=LA, $P$=LA, $Q$=SF
Goal State: $P$=LV, $Q$=LV

Actions:
- move($x$, $y$) = {prec: $[T=x]$, eff: $[T=y]$}
- loadP($x$) = {prec: $[T=x, P=x]$}, eff: $[P=Tr]$}
- loadQ($x$) = {prec: $[T=x, Q=x]$}, eff: $[Q=Tr]$}
- dropP($x$) = {prec: $[T=x, P=Tr]$}, eff: $[P=x]$}
- dropQ($x$) = {prec: $[T=x, Q=Tr]$}, eff: $[Q=x]$}

Plan: loadP(LA), move(LA, SF), loadQ(SF), move(SF, LV), dropP(LV), dropQ(LV)

How to construct such a formula?
How many actions are in a step?
(step = set of actions)

The foreach step
- The preconditions of all actions in a step must already hold in the beginning of the step
- The effects of all actions must hold at the end of this step
- The actions in a step do not interfere – they cannot destroy each others preconditions by their effects
- The actions in a step can be turned into a valid subplan sequence

The exist step
- The preconditions of all actions in a step must already hold in the beginning of the step
- The effects of all actions must hold at the end of this step
- The actions in a step do not interfere – they cannot destroy each others preconditions by their effects
- The actions in a step can be turned into a valid subplan sequence

The relaxed exist step
- The preconditions of all actions in a step must already hold in the beginning of the step
- The effects of all actions must hold at the end of this step
- The actions in a step do not interfere – they cannot destroy each others preconditions by their effects
- The actions in a step can be turned into a valid subplan sequence

Relaxed relaxed exist step
- The preconditions of all actions in a step must already hold in the beginning of the step
- The effects of all actions must hold at the end of this step
- The actions in a step do not interfere – they cannot destroy each others preconditions by their effects
- The actions in a step can be turned into a valid subplan sequence

Example: shortest plans for different semantics
- foreach: loadP(LA) → move(LA, SF) → loadQ(SF) → move(SF, LV) → dropP(LV) → dropQ(LV) → 5 steps
- exist: loadP(LA), move(LA, SF) → loadQ(SF), move(SF, LV) → dropP(LV), dropQ(LV) → 3 steps
- relaxed exist: loadP(LA), move(LA, SF), loadQ(SF) → move(SF, LV), dropP(LV), dropQ(LV) → 2 steps
- relaxed relaxed exist: loadP(LA), move(LA, SF), loadQ(SF), move(SF, LV), dropP(LV), dropQ(LV) → 1 step

Basic ideas of the relaxed relaxed exist step SAT encoding
- The SAT encoding only approximates the semantics, i.e., the satisfiability of the constructed formula $F_n$ implies the existence of a k-step plan (not vice versa)
- The actions are ranked using cycle-ignoring topological sorting on the action dependency graph (action ranking can be arbitrary as long as it is injective)
- The encoding allows only lower ranking actions before higher ranking ones in a step
- The encoding uses implication chains similar to those used in the exist step and relaxed exist step encoding

Conjectures
- Using a more relaxed semantics allows us to find plans with fewer steps
- Fewer steps means fewer SAT formulas to solve, which leads to finding plans faster

Experimental setting
- We compared 3 of the 4 encodings on eight International Planning Competition (IPC 2012) domains (20 problems each)
- All formulas were solved with the same SAT solver – Lingeling
- Computer: Intel i7 920 cpu @2.67 Ghz and 6 GB of memory
- The time limit was 30 minutes for a step
- We measured the number of problems that were solved within the time limit and the number of steps needed

Conclusion
- We have defined a novel parallel planning semantics and a SAT encoding which approximates it
- The results of the experiments show that the new encoding is successful in solving IPC benchmark problems
- For the domains Pegsol, Barman, and Visitall we achieved a significant improvement in the number of solved instances
- The average number of required steps decreased for all domains, most significantly for the Visitall domain
- The encoding can be further improved to produce smaller formulas and to better approximate the defined semantics