Summary: Generalized Planning with Projections and Trajectory constraints. 12/12/2016.

1. In B&G (the IJCAI 2015 paper), a policy μ was shown to generalize to (usually deterministic) problems P1 .. Pn, when A) all Pi's reduce to some abstract/smaller non-deterministic problem P', B) μ was a strong solution to P', and C) μ terminates on the Pi's.

   One of our goals now is to eliminate the condition C, possibly changing condition B as well by using *trajectory constraints.*

   In addition: it looks that we can often take the problem projected on the observations P_o as the reduced problem P': P_o, has the the observations as the possible states, and a transition (o, a, o') belongs to P_o iff there is a transition (s, a, s') in some P_i such that $o = o(s)$, $o' = o(s')$. This object, P_o, sounds like an object people would have studied before, especially in the context of abstraction. Does anyone know?

2. Trajectory constraints refer to formulas (in principle in LTL) that restrict the set of possible state trajectories. There are trajectory constraints that are true in each Pi that get "lost" in the reduced problem P'. The situation is like in *databases:* if we take the join of projections of a database, we don't necessarily get back the database. Some information may be lost. In our setting, the reduced problem P' contains the projection of the transitions and something similar occurs. This is why the termination condition C is needed; a condition that hopefully can be avoided by *making the trajectory constraints explicit.*

3. Example of a trajectory constraint C_D: if an integer variable, initially positive, keeps being decremented by 1 while being incremented a finite number of times only, eventually, the variable will have value zero. Something that in LTL could be expressed as:

   > if eventually var-not-incremented, and always eventually var-incremented, then eventually variable equal zero

4. Provided with a suitable set of trajectory constraints C (or set of trajectories .. defined over actions and observations), the notion of generalization could be expressed as:

   New Definition: DEF μ solves $P + C$ iff every state trajectory compatible with μ and P that satisfies trajectory constraints C, reaches the goal of P. Remark: this is equivalent to saying that every compatible trajectory satisfies $C \rightarrow \mathbf{F}T$.

   That is: this notion of "solves" doesn't assume fairness (as in strong cyclic solutions); instead, it requires reaching the goal (weakly) in the trajectories that are compatible with the constraints.

   New Generalization: THM If μ solves $P + C$, and Q reduces to P and satisfies C, then μ solves Q.

5. We showed a "completeness" result, where roughly if μ solves a possibly infinite collection of instances $P_1, \ldots, P_n, \ldots$ all of which reduce to P, and C is taken to represent the action-observation sequences that result from at least one of these instances, then μ will solve $P + C$ (i.e., P given C).

   I recall we took P to be P_o and targets to be visible, i.e., $T \subseteq O$:

   We are given a set $\mathcal{P} = \{P_1, P_2, \ldots\}$ of POND over the same set O of observations and A of actions. Define the "existential abstraction of $\mathcal{P}$" to be the POND P_o whose states are O, actions A, and transitions (o, a, o') if there exists i and $s, s' \in P_i$ such that $obs_i(s) = o, obs_i(s') = o'$ and $(s, a, s') \in \delta_i$. If $\pi = s_0 a_0 s_1 a_1 \ldots$ is a trajectory in some P_i, write $obs(\pi) = obs(s_0) a_0 obs(s_1) a_1 \ldots$ for its observation sequence. Let C be the set of observation sequences $obs(\pi)$ of trajectories π of the $P_i s$. Then: a policy μ solves $P_o + C$ iff it soves every P_i.

   Indeed, suppose it solves $P_o + C$ and let π be a trajectory of some P_i. Its observation $obs(\pi)$ is a trajectory of P_o (by dfn of P_o) that, moreover, satisfies C (by dfn of C). Thus $obs(\pi)$ reaches the target (since the policy is assumed to be successful on $P_o + C$) and since targets are visible, π reaches the target.

Conversely, suppose the policy solves every P_i. Let $\rho = o_1 a_1 o_2 a_2 \ldots$ be a trajectory of P_o that satisfies C. By dfn of C, $\rho = obs(\pi)$ for some trajectory π of some P_i. Thus π reaches the target (since the policy solves P_i). Since the target is visible, some $o_j \in T$. Thus ρ reaches the target.

6. Two questions here:

   Q1 What are the trajectory constraints C needed in concrete examples?

   Q2 How to compute policies μ that solve $P+C$; i.e., policies that solve a non-det problem P *assuming* the trajectory constraints in C but no "fairness" as in strong cyclic planning?

7. **About Q1:** We looked at wall-following and definitely we want to get there, but it's not the best starting point, as the "goal" of the problem is a bit complex (in the paper it is expressed as finding a prize hidden in a cell close to the bottom "landscape"; not too nice, needs to be worked out). There are also problems where a hidden object has to be found in a rectangular grid, and the visual marker where marker is supposed to be placed on a green block, etc etc. We should get there, but we can start with simpler problems.

8. *Problems with actions that increase and/or decrease subsets of integer non-negative variables, and possibly some booleans.* For example, the problem of getting X to 0 given an action with precondition $Y = 0$ that decreases X and increases Y, and another action that decreases Y, can be solved with two nested loops: decrement Y until $Y = 0$, decrementg X when $Y = 0$ until $X = 0$ (btw: while the IJCAI 2015 paper, assumes no action precondition, it's direct to add observable preconditions).

9. This type of problems with counters that can be decreased and increased is considered in Srivastava et al, AAAI 2011. These problems are solved in two steps: strong cyclic policies are computed (assuming that DEC actions on a positive variable may leave the var positive or make it zero), and these policies are then shown to terminate (notice the similarity with our IJCAI 2015 results, where we strong-cyclic solve the "reduced" problem P and show then that the policy terminates in the target problems P_i that reduces to P). Termination is shown with a so-called *sieve algorithm:* inner loops are labeled as "terminating" and hence removed, when there is a variable in the loop that is decreased and is not increased. This continues until outer loop is labeled as terminating.

10. As a first step, I think that we can capture such Increment/Decrement policies over a set of integer-valued vars with the trajectory constraints C_D above, implicitly used in the Sieve algorithm; namely that "if eventually a var is no longer increased, and always it is eventually decreased, then eventually it will have value zero". That is, a policy that solves the non-det problem $P+C_D$ (P given the trajectory constraints C_D), will solve any problem P' that reduces to P and satisfies C_D without appealing to the notions of "strong cyclic solutions" or "termination". Just to DEF above.

11. **About Q2:** One approach to compute policies that solve $P + C$ where C is a set of trajectory constraints expressed as LTL formulas, is by appealing to LTL synthesis algorithms. Also, if P is non-det and C maps into a DET Buchi Automata, the problem has been addressed, e.g., in Patrizi et al, IJCAI 2013 (in the full observable case). We need to provide a general algorithm for that but, as interesting, is to come up with an effective approach ...

    Yes, if C is given as an LTL formula one can use LTL synthesis (under perfect information):

    Suppose C is given by an LTL formula ψ over alphabet $O \cup A$, where O is the set of observations and A the actions, assumed disjoint. Form the LTL formula $\varphi = \psi \to \mathbf{F}T$ (where $T \subseteq O$ is the given visible target set). Convert φ into a (UCW) universal co-buchi word automaton W, i.e., the language accepted by W is exactly the set of models of φ; build a UCT (universal co-buchi tree automaton) T from W that accepts those O-branching A-labeled trees, representing tree-unfoldings of systems P, such that every trajectory of P satisfes φ. The system P is accepted by UCT T iff $P + C$ is solvable.

    The cost (in time) of building the UCW W is $exp(|\varphi|)$. The cost of building the UCT T is $exp(|W|)$. The cost of testing membership of P in T is polynomial in T. Thus, the cost of solving the POND $P + \psi$ is $exp(exp(|\psi|))) \times poly(|P|)$, i.e., 2exp in $|\psi|$ and polynomial in P.

This also shows that if C is given as an NBW then was save one exponent, i.e., the cost is $exp(|\psi|))) \times poly(|P|)$. Moreover, if C is given as a DBW then we can directly build a deterministic co-buchi tree automaton T, thus we get cost $poly(|\psi| \times |P|)$.

12. **Compiling trajectory constraints away:** A different computational approach is to "map" the trajectory constraints C into "fairness" constraints, so that the compiled problem can be solved by "strong cyclic planners". For example, the DECrement action with the trajectory constaints C_D that says that "if the var keeps being decremented and is not incremented, then it'll eventually have value zero", can be compiled into a new DEC action with the same two possible outcomes (variable stays positive or is zero), and **extra precondition** NO-INC that makes the action **fair:** namely, NO-INC is a boolean, that is initially true, and that is made false by any action that INCrements the variable. In other words, extra atoms are introduced in the problem to track that certain conditions remain true/false, and they are used to guarantee that, given the trajectory constraints, the action will be fair; i.e., it will not skip forever some of the outcomes. This needs to be made formal and precise but the general aim is to go back to **strong cyclic planning without trajectory constraints, using the trajectory constraints and additional preconditions to ensure that the assumption of "fairness" underlying strong cyclic planning holds.**

13. **Next 1:** If this is roughly correct, then possible next steps: 1) see if this "compilation" approach is sound and complete wrt to these INC/DEC counters problems (with actions potentially increasing/decreasing more than one variable). 2) see if this will suffice for dealing with problems that apparently don't have this form: like Wall-following, Visual Marker, Prize, etc etc. 3) Ultimately: it'd be nice if a **domain** could be characterized in compact form, not by action schemes and/or a potentially infinite set of instances, **but** by a "reduced (propositional) problem P" and a set of trajectory constraints. That is, the domain represents all the problems that reduce to P and that satisfy the trajectory constraints. For some domains, this may be feasible, for others, it may not ...

14. **Next 2:** Organizing this all, and thinking in terms of a paper: the first part should should be basic and theoretical, introducing the necessary definitions to state and proof the two basic properties (that looks that we have by now):

    - **Soundness:** That if A) $P_1, \ldots, P_n$ is a family of problems that reduce to P, B) C is a set of trajectory constraints true in all P_i's (or a sound set of possible action-observation sequences), and C) μ solves P given C, THEN μ solves each P_i.
    - **Completeness:** That if A) $P_1, \ldots, P_n$ is a family of problems that reduce to P, and B) μ solves each P_i, THEN there is a set of trajectory constraints C true in all P_i's such that μ solves the observation projection P_o given C (from the definition, every P_i reduces to P_o)

    Then the rest, should be about illustrating these results over different examples/domains; and about computational approaches for solving $P + C$ (P given C). The first includes illustrations of trajectory constraints for various examples, and the second includes the compilation of $P + C$ into a problem P' that can be solved by a strong cyclic planner (hopefully) and/or some other ideas (some general algorithm off-the-shelf, not necessarily "practical", for workign directly on the $P + C$ representation, is needed as well).