Safe Goal-Directed Autonomy and the Need for Sound Abstractions

Siddharth Srivastava
School of Computing, Informatics and Decision Systems Engineering
Arizona State University
Tempe, AZ 85282
siddharths@asu.edu

Abstract

The field of sequential decision making (SDM) captures a range of mathematical frameworks geared towards the synthesis of goal-directed behaviors for autonomous systems. Abstract benchmark problems such as the *blocks-world domain* have facilitated immense progress in solution algorithms for SDM. there is some evidence that a direct application of SDM algorithms in real-world situations can produce unsafe behaviors. This is particularly apparent in task and motion planning in robotics. We believe that the reliability of today's SDM algorithms is limited because SDM models, such as the blocks-world domain, are *unsound* abstractions (those that yield false inferences) of real world situations.

This position paper presents the case for a focused research effort towards the study of sound abstractions of models for SDM and algorithms for efficiently computing safe goal-directed behavior using such abstractions.

Introduction

The increasing maturity of AI techniques presents us with a unique opportunity to develop physical and electronic AI agents that could autonomously assist humans. Such agents would need to be able to accept high-level commands, and reason about what to do over extended periods of time spanning multiple decision epochs. The field of sequential decision making (SDM) captures such problems. In order to solve them, an AI agent needs to the assess different courses of action available to it: which course of actions would accomplish the assigned task? would it be safe to execute? which course of actions would be beneficial? Evaluating a possible courses of action in this way requires some knowledge about the environment and the possible impacts of the agent's actions in it—in other words, *a model*.

In the absence of a model, such evaluations would need to be done through trial and error. It is difficult to conceive of situations where deploying robots would have a high value and where such trials and their associated errors would be acceptable. In situations that involve proximal human-robot collaboration, or situations that are too dangerous for humans, errors are usually associated with forbidding penalties. Just as a bomb-disposal robot that learns on the fly would be an ephemeral investment, a household assistant that attempts to learn through trial an error, which medication is required when a person goes into insulin shock, would be of dubious ethical, social and economic value. It is well known in the AI community that PAC-learning guarantees alone are not sufficient for ensuring safe behavior in such situations; recent analyses have highlighted their limitations in the face of the anticipated roles of AI systems (Russell, Dewey, and Tegmark 2015; Brynjolfsson and Mitchell 2017).

The focus of this position paper is on the *mechanisms for creating domain models that are efficient but sound abstractions of real-world problems, and the algorithmic advances required for using such models for safe behavior synthesis*. Models can be in the form of closed-form mathematical specifications, (such as Markov Decision Process with transition probability specifications) or in the form of black-box simulators or generative models that can sample possible action outcomes (as typically used in reinforcement learning). Models of either form can be derived from existing knowledge, or learned through past experience in the field. Indeed, some of the most popular demonstrations of AI systems rely upon *perfect models* (Silver et al. 2017; Mnih et al. 2015) in the form of game simulators for efficiently obtaining millions of labeled behavioral experiences.

Regardless of the form or the nature of acquisition of models, higher fidelity models feature larger branching factors and larger time horizons and therefore result in SDM problems of higher computational complexity (regardless of the solution approach taken, be it dynamic programming, search, learning from trials and past experience, or a combination thereof). Hierarchical abstractions are used to alleviate this problem by creating input models that are abstractions of the true problem (Sacerdoti 1974; Knoblock 1990; Parr and Russell 1998; Dietterich 2000; Marthi, Russell, and Wolfe 2007).

Hierarchical abstractions include state abstractions (models that maintain fewer environment properties than the real situation) as well as temporal abstractions (models featuring high-level actions that span multiple primitive operations of the underlying actuators).

In recent work (Srivastava, Russell, and Pinto 2016) we showed that simple forms of abstractions can result in models that are not consistent with the underlying problem scenario as well as models that are not Markovian, or not solvable! As a result many real-world problems have never truly been addressed by SDM solution techniques that treat their input models as perfect abstractions.

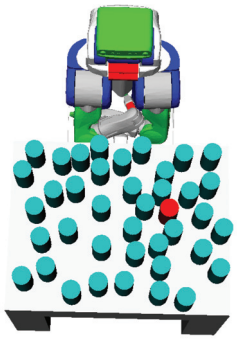

Figure 1: A realistic blocks-world problem. Pickups can be made only from the sides. Although there is no stacking and the preconditions of the pickup action are satisfied, there is no feasible motion plan for picking up most of the objects on the table.

For instance, consider the blocks-world domain, which is among the most easily recognizable, perhaps even infamous, benchmarks for sequential decision making. Given initial and desired configurations of blocks on a surface, the problem is to compute a behavior that would transform the initial configuration to the desired configuration. The set of available actions typically consists of maneuvers such as *pickup* and *place*. Stochastic action effects and noisy sensors for this domain can be easily expressed in most modeling languages for SDM (Boutilier, Reiter, and Price 2001; Younes and Littman 2004; Sanner 2010; Srivastava, Cheng, and Russell 2014). Although SDM *models* for the blocks-world domain are considered to be too "well studied" to be interesting for research, they are poor abstractions of the underlying SDM *problems* of rearranging objects while avoiding collisions (see Fig. 1 for a simplified yet realistic problem). Consequently the underlying problems remain unsolved and feature significant research challenges.

Indeed, while the true space of blocks-world problems captures all pick-and-place problems, ongoing research in robotics shows that SDM solvers that perform well on the standard blocks-world model produce poor *unsafe* solutions even in simplified real-world situations that feature robots with perfect sensing and actuation, and block arrangements without stacking (Cambon, Alami, and Gravot 2009; Kaelbling and Lozano-Pérez 2011; Plaku and Hager 2010; Kaelbling and Lozano-Pérez 2013; Srivastava et al. 2014). These solutions could violate arbitrarily many of the constraints that were abstracted in an unsound fashion, and result in unsafe behaviors that include unintended collisions. This situation is representative of several problems where goal-directed autonomy is desired; we believe that this potential for unsafe behavior effectively prohibits the safe deployment of general purpose AI agents.

Problem with the Current Situation

Conventional modeling paradigm As noted above, it is well appreciated that abstraction is a useful mathematical tool for solving real-world SDM problems. The conventional wisdom along these lines is to use an abstract domain model with an SDM solver to compute the "high-level strategy" for solving a problem (e.g., one that determines the order of unstacking and stacking block-tower configurations), and then use a low-level planner (e.g., a motion planner) or controller to implement each of the actions in the strategy.

Underlying assumptions and their limitations This wisdom is based on the assumption that the effect of applying an action in the real world will be consistent with the modeled effect in the abstract model. This in turn is based on the assumption that the result of applying a desired abstraction function on the real situation will be a Markovian model.

On the other hand, constructing a Markovian model requires the inclusion of several properties of the environment as state variables or predicates; abstraction requires *removing, or coarsening properties* in the model. It should therefore be natural to expect the abstraction of an accurate domain model to possibly result in a *non-Markovian domain model*. Recent work shows that this intuition is in fact true (Srivastava, Russell, and Pinto 2016): simple abstractions can result in models that are not Markovian; furthermore, it is often not possible to express the resulting models accurately in existing modeling languages for SDM.

This raises a few questions: all the SDM models we use are Markovian (and naturally, are expressed in the modeling languages that we have been using). Few, if any, of these are accurate, non-abstracted depictions of the real world situation that they represent. Have we been lucky enough to always get Markovian abstractions? Do the domain designers intuitively construct perfect abstract transition systems that retain just the right properties to make the resulting abstracted model tractable as well as Markovian?

To answer these questions, we turn once again to the blocks world and its abstraction expressed as the blocks-world domain. Among other details, this domain states that if a block has nothing on top of it, the robot's gripper should be able to pick it up. In a real situation (e.g. Fig. 1), this is *not true* because there may be no collision-free path for the gripper to pick up the block. The vocabulary used in the blocks-world domain is not sufficient to accurately express this property (Cambon, Alami, and Gravot 2009; Hertle et al. 2012; Kaelbling and Lozano-Pérez 2011). *As a result the standard blocks-world domain is not a sound model of the real blocks world because it implies action consequences that are not true* [1]. Policies computed using such models are unsafe, and can be dangerous. Although our example refers to situations where geometric constraints were abstracted out, such errors can arise with all forms of abstraction. One would not appreciate a robot using such principles in most applications that could benefit from a safe and productive robot, including mining, firefighting, bomb disposals, household help, etc.

[1] It is sound for environments where the gripper is either infinitesimally thin (so that it can slip between adjacent towers), or is an electromagnet suspended from the ceiling. Either way, the ceiling should be arbitrarily high and the table should be broad enough to lay any number of blocks on it. Such situations are unusual if not impossible.

In fact, the sound abstraction of the blocks world using the vocabulary of the blocks-world domain is a non-Markovian transition system: the effect of reaching for a block in this transition system depends on the occurrence of preceding *place* actions. If the target block was initially reachable, and no other *place* actions placed a block on the same table, the block will remain accessible. Otherwise, it may not be. *Therefore, the standard blocks-world model is not only inconsistent with the underlying problem, its vocabulary is insufficient to make the abstract transition system Markovian!* Forcing such a non-Markovian abstract transition system into domain languages that can only express Markovian models results in a model that is inconsistent with the modeled problem. Our research indicates that the situation can be resolved if the modeling languages are extended to annotate parts of the model as imprecise due to abstraction, and algorithms utilize this information to extract more information from higher fidelity models when needed.

Non-solutions The preceding discussion may *seem to indicate* that a stochastic formulation (such as an MDP) would help resolve these issues. However, this is not true. First, it would require enumerating and solving for the complete set of *possible* outcomes for an action in an abstract state space. This is infeasible. E.g., in the blocks-world model's vocabulary, every time a robot (not a ceiling mounted gripper) tries to move its hand, all possible subsets of movable objects in the room would need to be considered as potentially being knocked over. Second, such models would not be *complete*: they would disallow solutions that are feasible under a more accurate representation.

The problems highlighted above are orthogonal to efforts aimed at increasing the level of detail expressible in our input modeling languages (e.g., (Hertle et al. 2012; Fox and Long 2002)). Even if we could model SDM problems at the level of detail of sub-atomic particle interactions, this is unlikely to yield more efficient solution techniques. It is equally unlikely that modeling an entire household at the level circuit diagrams of every appliance would "help" a household robot efficiently compute useful behavior. Natural computational consequences of increasing branching factors and time horizons make it clear that a uniformly detailed model at the highest possible fidelity will not yield the most efficient SDM system, regardless of the solution approach. Thus, SDM solvers will continue to rely upon hierarchical abstractions for efficiency in modeling and in solution computation.

Paths Ahead

We believe that the limitations in correctly expressing abstract SDM models of real-world situations (and consequently, of efficiently solving such problems) have limited the applicability of SDM techniques in the real world. As a community we have made numerous advances under the assumption that inputs will be perfect abstractions that yield exactly the true consequences. Our position is that these advances are necessary, but not sufficient towards deployable autonomous systems. We also need to expand the scope of SDM technology towards principled approaches for designing and computing abstract SDM models that may be imprecise, but not incorrect. New representations for such abstract SDM models (generative models or *simulators*, as well as analytical) would require and facilitate corresponding algorithms that produce truly executable solutions.

Some prior research efforts are highly relevant to this problem. Work on algorithms for planning with models that may be incomplete addresses situations when unknown perturbations may have been applied to accurate domain models that are expressible in the modeling language (Nguyen and Kambhampati 2014). Angelic semantics for high-level actions increase the scope of representation languages to specify upper and lower bounds on reachable states in situations with temporal abstraction rather than state abstraction. The resulting algorithms are able to effectively utilize such bounds in pruning irrelevant high-level actions (Marthi, Russell, and Wolfe 2007). Related research in motion planning highlights the value of state abstractions of control-theoretic models, which are constructed using subsets of the full set of variables required to describe a system (Styler and Simmons 2017). We have been developing representations for efficiently expressing imprecise but sound abstract models resulting from state and temporal abstraction for arbitrary SDM problems. Our solution algorithms utilize sound and imprecise abstract models, but dynamically improve them by deriving abstracted, context-sensitive information from more accurate models. This information is abstracted and incorporated in the abstract models (Srivastava et al. 2014; Srivastava, Russell, and Pinto 2016), allowing SDM algorithms to compute agent behaviors with strong guarantees of safety and correctness. Some of our main results can be summarized as follows:

1. Under certain conditions, abstraction can indeed result in Markovian models. These conditions appear to be rare.
2. In many cases, abstraction results in domain models that includes forms of model-imprecision that could have been resolved during computation had they been expressed. However, current modeling languages do not support constructs that distinguish model imprecision arising due to abstraction from non-determinism or stochasticity that is a feature of the environment.
3. If model imprecision caused due to abstraction is recorded in the abstract model (e.g., by noting that the effect of a *place* action is imprecise, along with the abstraction function that caused the imprecision), the situation can be resolved. It is possible to dynamically tune the abstraction to include more information from accurate models using different solvers for models at different levels of abstraction. Used in this fashion, SDM solvers can effectively produce executable behavior. Dynamically tuning an imprecise (but not incorrect) model during search allowed us to produce a competitive task and motion planner that uses existing SDM solvers.

These initial results indicate that new methods for computing and utilizing abstract models that are sound even when they are imprecise allow us to leverage SDM technology towards solving entire new classes of problems that are abstractions

of real-world situations.

Acknowledgments

I thank Rao Kambhampati for helpful discussions and comments on this paper.

References

Boutilier, C.; Reiter, R.; and Price, B. 2001. Symbolic dynamic programming for first-order mdps. In *Proc. IJCAI*, volume 1, 690–700.

Brynjolfsson, E., and Mitchell, T. 2017. What can machine learning do? workforce implications. *Science* 358(6370):1530–1534.

Cambon, S.; Alami, R.; and Gravot, F. 2009. A hybrid approach to intricate motion, manipulation and task planning. *IJRR* 28:104–126.

Dietterich, T. G. 2000. Hierarchical reinforcement learning with the maxq value function decomposition. *J. Artif. Intell. Res.(JAIR)* 13:227–303.

Fox, M., and Long, D. 2002. PDDL+: Modeling continuous time dependent effects. In *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*.

Hertle, A.; Dornhege, C.; Keller, T.; and Nebel, B. 2012. Planning with semantic attachments: An object-oriented view. In *Proc. ECAI*.

Kaelbling, L. P., and Lozano-Pérez, T. 2011. Hierarchical task and motion planning in the now. In *Proc. ICRA*.

Kaelbling, L. P., and Lozano-Pérez, T. 2013. Integrated task and motion planning in belief space. *The International Journal of Robotics Research* 32(9-10):1194–1227.

Knoblock, C. A. 1990. Learning abstraction hierarchies for problem solving. In *Proc. AAAI*.

Marthi, B.; Russell, S. J.; and Wolfe, J. 2007. Angelic semantics for high-level actions. In *Proc. ICAPS*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Nguyen, T. A., and Kambhampati, S. 2014. A heuristic approach to planning with incomplete STRIPS action models. In *ICAPS*.

Parr, R., and Russell, S. J. 1998. Reinforcement learning with hierarchies of machines. In *Proc. NIPS*.

Plaku, E., and Hager, G. D. 2010. Sampling-based motion and symbolic action planning with geometric and differential constraints. In *Proc. ICRA*.

Russell, S.; Dewey, D.; and Tegmark, M. 2015. Research priorities for robust and beneficial artificial intelligence. *AI Magazine* 36(4):105–114.

Sacerdoti, E. D. 1974. Planning in a hierarchy of abstraction spaces. *Artificial intelligence* 5(2):115–135.

Sanner, S. 2010. Relational dynamic influence diagram language (rddl): Language description. http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; Chen, Y.; Lillicrap, T.; Hui, F.; Sifre, L.; van den Driessche, G.; Graepel, T.; and Hassabis, D. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676):354–359.

Srivastava, S.; Fang, E.; Riano, L.; Chitnis, R.; Russell, S.; and Abbeel, P. 2014. A modular approach to task and motion planning with an extensible planner-independent interface layer. In *Proc. ICRA*.

Srivastava, S.; Cheng, X.; and Russell, S. 2014. First-order open-universe POMDPs: Formulation and algorithms. In *Proc. UAI*.

Srivastava, S.; Russell, S.; and Pinto, A. 2016. Metaphysics of planning domain descriptions. In *Proc. AAAI*.

Styler, B. K., and Simmons, R. 2017. Plan-time multi-model switching for motion planning. In *Proc. ICAPS*.

Younes, H. L., and Littman, M. L. 2004. PPDDL 1.0: An extension to pddl for expressing planning domains with probabilistic effects. *Technical Report CMU-CS-04-162*.