Intelligent Narrative Technologies and Social Believability in Games:
Papers from the AIIDE 2015 Joint Workshop

Implementing Injunctive Social Norms Using Defeasible Reasoning

Joseph A. Blass and Ian D. Horswill
Northwestern University
joeblass@u.northwestern.edu; ian@northwestern.edu

Abstract

Believability requires video game characters to consider their actions within the context of social norms. Social norms involve a broad range of behavioral defaults, obligations, and injunctions unrelated to strictly causal reasoning. Defeasible reasoning involves rationally compelling but deductively invalid arguments, such as reasoning with rules that allow exceptions. This paper investigates having video game characters use defeasible reasoning to consider injunctive social norms when selecting and planning actions.

Introduction

Action selection and planning involves selecting policies, actions, or sequences of actions that are likely to further an agent's goals (Ghallab et al., 2004). This is a difficult enough problem, but an additional challenge for designers of believable virtual characters is having these characters act in accordance with social norms. Social norms involve a broad range of behavioral defaults, obligations, and injunctions that are orthogonal to the strictly causal reasoning generally implemented in planners. These norms can range in significance from the gravely serious, such as moral injunctions against murder, to the almost trivial, such as the convention of people passing one another on the sidewalk bearing to the right. Norms enable us to consider the social acceptability of an action, and draw our attention to socially relevant information, such as who is going to be affected the most by the action. Norms define what actions in what contexts are suggested, obligatory, or forbidden, and allow multiple agents to co-exist and cooperate without explicit coordination. Failure to follow norms can lead to character behavior that is comical, obnoxious, or simply strange, in the eyes of a human observer.

In principle, social norms could be encoded directly into the axiomatization of a domain by salting planning operators with additional preconditions and effects, and adding additional goals such as "don't be a jerk." However, there are a number of reasons why this might be undesirable. Modularity argues for centralizing representation of a phenomenon where possible, rather than diffusing it through all the various planning operators. More importantly, social norms have a fundamentally different character from the causal structure of a domain. Causal rules, such as that firing a gun requires the gun be loaded, are generally inviolate. Social norms on the other hand are nearly always *ceteris paribus* rules that admit endless exceptions and excuses. Moreover, social norms often conflict with one another, requiring case-by-case arbitration as to which norms win in a given circumstance.

In this paper, we discuss initial work on the integration of reasoning about social norms into a character agent architecture in the game MKULTRA (Horswill, 2014) using defeasible reasoning (reasoning with exceptions). We will describe the system for reasoning about norms, its integration into the overall character architecture, and the results of early experiments with the system.

It should be said at the outset that social norms are a very broad area, and our work does not cover all cases. Norms can involve overt obligations for action, such as greeting someone at the beginning of a conversation, or overt injunctions against it, such as not stealing someone else's property. However, they can also involve much more subtle issues such as when a contemporary American male tries to determine whether he knows another male well enough to greet him with a hug rather than a handshake. In this paper, we deal only with overt injunctions, and the reasoning about their exceptions.

Social Norms in Game Characters

Social norms commonly occur in social interactions and actions that affect others. They are relatively unimportant in games in which NPCs serve as violent opponents or helpful sidekicks to a player character, rather than a social partner. These roles demand less social intelligence since they afford little by way of unscripted social interactivity. However, several games have implemented systems to handle and respond to social norms, which we will discuss.

In *Façade* (Mateas & Stern, 2003), NPCs could deem an action on the part of the player character, such as kissing one of the characters in a married couple, inappropriate, and respond accordingly. NPCs were written in such a way as to not produce inappropriate behavior themselves, but to our knowledge, they did not include any sort of reified system of social norms.

The Sims 3 (Evans, 2009) used a propositional rule engine to reason about character actions and responses. Rules were used both to filter inappropriate character actions, and to recognize actions as inappropriate. This allowed the implementation of an "inappropriate" personality trait that could be applied to a given character to make them intentionally behave inappropriately in social situations.

Evans (2010) also used a rule-based system for a deontic logic to implement explicit reasoning about social norms in an unpublished prototype Sims-style game. In the game, norms could be stated as a deontic rules and changed dynamically during the game.

Versu (Evans & Short, 2014) implements complex social interactions using a reified model of social practices. In Versu, social practices are effectively parallel processes that run independently of the characters participating in them. They propose actions to the characters, who then determine which actions to perform by forward-simulating the actions, then scoring the resulting world states based on a set of goals. Goals can be arbitrary sentences in its logic. Norms can be explicitly modeled by adding goals to the characters to not violate them. This makes for an extremely flexible system, but only allows for single-action lookahead.

There are two components missing from these previous models of social reasoning for virtual characters in video games. The first is the ability to have norms override other norms. For example, the inclination not to touch strangers is outweighed by the obligation to help people when someone falls down in the street, which is overruled by a different norm if the person who fell has several bodyguards. The second missing component is the ability to integrate conflicting information. Social practices are highly confusing and confusable: some information (or agents) can indicate that one action is correct, while others indicate another. This is not simply a matter of taking a weighted sum, but of explicitly coordinating and integrating conflicting inputs.

Defeasible Reasoning

Defeasible reasoning (Pollock, 1987) involves argument that is rationally compelling but not deductively valid (Koons 2014). In practice, this generally means reasoning with rules that allow exceptions (*ceteris paribus* rules). This makes it an appealing, if somewhat legalistic, tool for reasoning about social norms, since it provides a convenient formalism for expressing both *ceteris paribus* rules, and their exceptions.

Defeasible reasoning, in the form of non-monotonic logic, has received a great deal of attention in the knowledge representation community, beginning with McCarthy's work on circumscription (1980), and continuing with work on various logics such as Reiter's Default Logic (Reiter, 1980) and Moore's Autoepistemic Logic (Moore 1985). We use here Nute's (1993) Defeasible Prolog (d-Prolog), a meta-interpreter written in Prolog that supports a subset of pure Prolog together with extensions for defeasible Horn rules. D-Prolog has been used previously to model semi-formal norms such as lending policies (Ryu 1992) and parking regulations (Dhanesha 1994).

Defeasible reasoning provides a path towards integrating conflicting information and letting certain norms trump others. Defeasible reasoning lets characters reason with a set of defaults, but allows those defaults to be overridden by other rules in unusual situations or when circumstances change. This makes it a good choice for reasoning about social norms.

The key idea behind defeasible reasoning is that not all reasons, which link arguments together, logically entail their conclusions. Instead they create a defeasible "presumption in favor of their conclusion" (Pollock, 1992), that is, a reason to assume the conclusion is correct until proven otherwise. In Pollock's formulation, conclusions are rejected by rules called *defeaters*, which either attack the *reasons* for a conclusion, or the conclusion itself. A rule which attacks the conclusion is a *rebutter*; a rule which attacks the reason for a conclusion is an *undercutter*. If something is undercut, the conclusion could still be true, but perhaps not for the reasons provided.

For example, let us imagine a rule that states, "defeasibly, if it's nice outside and Bill has a ball, Bill will play with a ball outside." Let us further assume that it's nice outside and Bill has a ball; we can conclude "presumably, Bill is playing with a ball outside." An undercutter would be a rule that states "assume it's not nice outside," detracting from our reasons for believing Bill is playing outside with a ball. A rebutter would be a rule that states "defeasibly, if Bill has a ball and the ball is a ping-pong ball, Bill will not play with a ball outside." Without knowing whether the ball is a ping-pong ball or not, we cannot know what to conclude.

Defeasible reasoning expands the range of query responses from "true" and "false" to "definitely true", "definitely false", "presumably true", "presumably false", and "can't tell", depending on how the conclusions were derived. It should be clear that, particularly when reasoning about the appropriateness of social behavior, a system that can ambivalently fail to draw a conclusion (that is, which concludes "I can't tell what the right thing to do is") is superior to one that cannot fail as gracefully. This is especially true if we want to be able to model the social dynamics of teenagers.

Defeasible reasoning was central to the OSCAR project, a cognitive architecture for rational agents that uses deductive inference rules and defeasible reasoning schemas as central cognitive inference tools (Pollock, 2000; 2001). OSCAR was designed to help make decisions and process information, rather than achieve goals. OSCAR is able to reason about a changing world using simple constraints. For example, if the agent perceives a thing P, it defeasibly believes P; if it does not subsequently perceive ~P (that is, not P), it continues to defeasibly believe P (Pollock, 1998). OSCAR can also project beliefs into the future and has ways of discounting those beliefs.

Defeasible Prolog (d-prolog, Nute 1993) is an extension to Prolog that implements defeasible logic. D-prolog adds operators to prolog that enable prolog to conclude the negation of a fact; to conclude something defeasibly; and to undercut a conclusion (no further operators are required to enable rebutting rules, since a rule that concludes the negation of a fact rebuts rules that conclude that fact). D-prolog also adds facts to support those new operators, such as declaring particular sets of facts incompatible or determine which rules are superior to others. D-prolog also enables assumptions: `itsNiceOut(today) := true.` means "presumably, we can conclude that it's nice out today", rather than traditional Prolog's `itsNiceOut(today).`, which means "it is definitely nice out today." D-prolog can derive conclusions strictly or defeasibly. Strict derivations only use regular Prolog rules; defeasible derivations use all rules. The use of the `:=` operator, rather than the Prolog's normal `:-` operator, states that a rule is defeasible, not a strict rule. It can be defeated by other rules that state exceptions or extenuating circumstances that rebut or override the original rule.

Implementation

The present work was implemented within MKULTRA, a game under development that incorporates compositional natural-language dialogue and reactive planning techniques in a tile-based RPG (Horswill, 2014). The game's reasoning system is written in Prolog, where each character has access to a global knowledge-base (KB) for the game as well as their own KB that encodes their beliefs, needs, goals, desires, etc. The natural language system uses definite clause grammars (Pereira and Shieber 1987) and is based loosely on CHAT-80 (Warren and Pereira 1982). The reactive planner is based on Sibun's (1992) Salix system, which is in turn based on McDermott's (1978) NASL system.

The current work focuses on injunctions, that is, on blocking potential actions that violate social norms. This involves two primary issues, reasoning about whether a given action is non-normative, and integrating that reasoning into the overall agent architecture. We use the term non-normative since it is more inclusive of the variety of norms than a term like forbidden. Some norms, such as moral rules, concern obligatory or forbidden actions, but other norms, such as the convention of passing on the right, are not inviolable. Both define normative and non-normative behavior, and the same framework can be used to reason about both kinds of norms. In the current implementation, however, the planner does functionally treat non-normative actions as forbidden: characters can only take a non-normative action if a different norm states that the action is actually normative.

Testing actions against norms

Reasoning about whether an action is non-normative is implemented using a set of defeasible rules written in d-Prolog. We will focus here on reasoning about object possession, although the system also includes injunctions against murder (with an exception for self-defense).

The primary rule used for object possession is that it is rude to use objects that belong to others:

```
rude(use(User, Obj)) :=
   belongs_to(Obj, Owner),
   User \= Owner.
```

Thus, it is rude for someone visiting a friend to walk up to their refrigerator and start eating their burritos. Characters are defined as belonging to themselves, so this rule also prevents a character from using another character for some purpose. This rule can be defeated by other rules that state exceptions or extenuating circumstances that rebut or override the original rule. One such rule is that it's not rude to use the object if one has permission to use it:

```
~rude(use(User, Obj)) :-
   permission_to_use(User, Obj).
```

Here the ~ operator denotes (strong) negation. This rule uses the :- operator and so is a strict (non-defeasible) rule:

it can't be overridden, and so rebuts the original defeasible rule.

Other rules can also override the original rule. For example, it might be acceptable to eat a friend's burrito without permission if one is literally dying of hunger:

```
~rude(use(User, Object)) :=
   satisfies_need(Need, Object),
   need_to_satisfy_to_survive(Need),
   satisfaction_level(Need, Level),
   minimum_survival_level(Survival_Level),
   Level =< Survival_Level.
```

However, this is another defeasible rule because it too can have exceptions, such as if one's friend is also dying of hunger.

Note that representing the full range of all human social norms would require an enormous number of rules, exceptions, exceptions to exceptions, etc. Clearly humans are able to store and reason with the enormous number of norms and conventions we are exposed to, but it would be too computationally expensive for us to do so exclusively with defeasible reasoning. That is, in this paper we are not making an argument that *humans* do all of our normative reasoning using defeasible reasoning. Scaling our characters' normative reasoning to human levels would be an enormous, perhaps impossible, challenge. However, defeasible reasoning provides the expressiveness required to begin exploring these issues in video game characters.

Architectural integration

Norms are integrated into the system's reactive planner by testing candidate actions for deviancy and rejecting ones that violate norms. The current system does not address the issue of taking additional actions, such as asking permission, to make otherwise deviant actions acceptable, although this could be done straightforwardly by declaring normativity to be a precondition for all possible actions and declaring other actions (such as asking permission) to establish normativity.

In a planner/executive architecture, norm testing would be integrated by filtering candidate actions in the planner. This would be easiest to do in a planner that used forward-chaining, state-space search, such as SHOP (Nau et al. 1999), since the full state of the world would be available for the norms reasoning system. However, it could also be used with other planners, such as partial order planners, particularly if the rules governing norms do not depend on state information that can be changed by plan actions.

In our implementation, characters use a reactive planner based loosely on Sibun's Salix (1992), which was in turn based on McDermott's NASL (1978). The system works by incrementally decomposing tasks until a primitive action is obtained, then executing the action, and continuing with the rest of the decomposition.

Domain knowledge is provided in the form of task decomposition rules. When executing a task, the system finds all possible decompositions of the task. If there is a unique decomposition, it executes it. If not, there is an impasse, and the system recursively executes either the task `match_failure(OriginalTask)` or the task `resolve_conflict(Task, Decompositions)`, depending on whether there were no decompositions or conflicting decompositions. These, in turn, have their own decomposition rules that implement meta-level reasoning for different tasks. This mechanism is similar to Soar's universal subgoaling (Laird et al., 1987).

The current system implements social norms by filtering candidate decompositions to veto decompositions that violate norms. Since this is a reactive planner (i.e. it commits to a particular decomposition and doesn't backtrack the execution of actions if the decomposition fails), it is possible for a character to commit to a decomposition and only later discover (when its subtasks are themselves decomposed) that it involves actions that would violate norms. In this case, the character would be forced to abort and restart the task using a different strategy. However, it might not do so until the original strategy had been partially executed. This could be alleviated either by adding additional domain knowledge to allow earlier rejection of deviant strategies, or by adopting a full (backtracking) planner.

One issue in the current implementation is that d-Prolog is implemented as a meta-interpreter, i.e. it is an interpreter written in Prolog that is itself interpreted by the underlying Prolog interpreter. D-Prolog queries are thus relatively expensive in the current implementation. This is partly due to features of d-Prolog that aren't used in the current implementation; for example, d-Prolog can evaluate queries relative to lists of explicit assumptions, a feature that is used for automatically determining some cases where one rule subsumes another. Our current implementation explicitly specifies rule precedence so as to avoid the overhead of repeatedly computing it.

In principle, this interpretation overhead could be compiled out through partial evaluation (Sterling and Shapiro, 1994). However, the current prototype uses an explicit list of action types that should be examined further for norm violations, with all other actions being automatically assumed permissible. This has so far proven to be relatively easy to maintain while also being efficient enough not to impact the frame rate of the game.

Example Scenario

We tested the system in a scenario where the player character visits a friend's house. Characters take objects to be owned by the owner of the house in which they reside, so all objects are assumed to belong to the friend. The characters run identical code bases, a Sims-style simulation of basic survival needs (Zubek 2010) that generates goals for the reactive planner discussed above (including the testing against social norms). The two characters thus have identical goals but differing behavioral injunctions, since one character owns all the objects, and the other owns none. The system was tested under two experimental conditions: in one, the friend declares that the player character should make themselves at home and do whatever they please, i.e. that they have permission to use anything in the friend's house. In the second condition, the friend declares that he is unhappy that the player character is at his house, and tells them not to touch anything. In this condition, the player does not get permission for any objects.

In the first condition, the two characters behave identically: they both go about fulfilling their needs. In the second condition, the friend goes about his day as normal, since he is in his house, owns everything, and therefore does not have to worry about whether he can use the items therein. However, the player character wanders and does (almost) nothing, since all actions involve either the friend or the friend's things, which is blocked by the injunction against using others' possessions without permission. Eventually, the player character's needs reach the point where her survival is in jeopardy (impending death due to hunger, dehydration, or sleep deprivation). At that point, the third rule above defeats the prohibition against using another's possessions without permission. This enables the character to fulfill her survival needs, but not other actions. She will therefore eat, sleep, and drink, but not watch television, for example, since television is not a survival need. Moreover, she only eats, sleeps, and drinks when those need levels near the fatal stage (i.e. below the emergency level).

A typical activity trace in this case is as follows. Suppose the emergency level for need satisfaction is 15 on a 0-100 scale, and the character's hunger satisfaction (i.e. level of satiation) is 20 and fun satisfaction is 10. Fun is not a survival need, and hunger is over 15, so the character does nothing. After sufficient time, hunger is below 15, and fun is almost 0. The character eats, which sets her hunger to 50. She then goes back to doing nothing until her hunger (or another survival need) satisfaction level dips below 15.

Other Related Work

In addition to work on socially believable characters in games, there is an interesting body of work on normative systems. Much of this work involves artificial multi-agent systems that use norms to define default behaviors in synthetic societies. Sergot (2008) presented a formal language for normative systems with which to discuss norms, compliance to said norms, and to determine which agents are responsible for the state of the world.

Dhanesha (1994) and Ryu (1992) have also used defeasible reasoning (and indeed, d-Prolog) to model semi-formal rule systems, such as parking regulations.

We are not aware of any work on normative systems that attempts to model existing human social norms, as would be desirable for simulated characters.

Future Work

This work is still in its initial stages. Most obviously, the system can be extended to support both more exceptions to the current norms and more types of injunctive norms.

As we add more norms, the system may have to reason about contradictory norms; for example, the norm of introducing oneself and offering a business card is of lower priority than the norm of rendering first aid to the victim of an automobile accident. Much work has been done on prioritizing rules in non-monotonic logics (i.e., Brewka & Eiter, 1999; Delgrande et al., 2003). D-prolog includes a facility for declaring relative priorities of rules, and even to make priorities conditional on the contents being reasoned about. D-prolog can also determine priority automatically in cases where one rule is derivable from a second, but the second is not derivable from the first (the second rule in this example has superiority). The current implementation does not involve enough rules to require much prioritization, however.

In the longer term, we will extend the system to more complicated types of reasoning. It would be useful for the system to be able to reason about the permissibility of sequences of actions, not just actions in isolation. For example, it is perfectly acceptable to go to dinner by oneself, and it is perfectly acceptable to invite someone to dinner, but not to invite someone and then stand them up. It would also be useful to be able to reason about permissibility in terms of the effects of actions rather than the actions themselves. Asking a houseguest to leave at the end of a dinner party is fine, asking them to do so during a dangerous storm might not be.

Characters should be able to have different norms from each other, and should be able to violate norms under certain circumstances. Currently characters can only violate one norm when another norm indicates that doing so is, in fact, normative. One simple solution is to have a different set of defeasible rules (which are not social norms) that define when it is reasonable for characters to ignore rules. Our current implementation is already capable of supporting characters having different norms, by having those norms reside in each character's KB instead of the game engine's KB.

Another area for future work concerns norms of obligation. If you're called away in the middle of a conversation, it's polite to first say "excuse me." On the other hand, if you're called away because a crazed gunman is shooting at you, you shouldn't waste time on pleasantries. We also do not currently represent norms surrounding omissions: what characters fail to do, not only what they have done. Making character architectures that can both conform to these norms, and reason about them (so as to feel slighted, for example, if someone walks away in the middle of a conversation) in a natural manner, is very much an open problem.

While the current system reasons about the normativity of actions, it would also be possible to implement a system that instead (or additionally) reasoned about the permissibility or impermissibility of states of the world. Our current system has only very limited support for this since our reactive planner does relatively little projection into the future. It can reason that eating another character is murder, and therefore immoral, because it causes to an impermissible state transition from living to dead. However, improved projection would certainly be useful.

Finally, there are a number of norms that do not conform to the kinds of quasi-legalistic reasoning used here. By quasi-legalistic reasoning, we mean reasoning that involves clear rules and exceptions with crisp evidence accruing in favor or against a conclusion. Some norms, for example the rules determining when one knows someone else well enough to use a nickname, are extremely complicated and do not necessarily fit the rules-and-exceptions model of defeasible reasoning. Determining whether a nickname is appropriate, for example, is more about reaching a limit point of personal connection than about a specific event. These are norms that depend on the accrual of many different small pieces of evidence and which can be satisfied in myriad ways, and which may not be effectively implementable using defeasible reasoning.

Conclusion

Conforming to social norms and conventions is critical to character believability. Developing agent architectures that can explicitly represent, reason about, and conform to such norms is an important goal for game AI research. Defeasible logic is a convenient tool for such reasoning because of its ability to naturally express complicated systems of rules and exceptions.

This work represents first steps in this project by adding support for representing and reasoning about injunctive norms and their various exceptions and mitigating circumstances. It shows that defeasible logic programming provides both an expressive representation and a reasoning system efficient enough for real-time control of game characters.

Acknowledgements

We gratefully acknowledge Richard Evans, Ken Forbus, and our reviewers for discussing and helping develop the ideas presented in this paper.

References

Brewka, G., & Eiter, T. (1999). Preferred answer sets for extended logic programs. *Artificial intelligence*, 109(1), 297-356.

Delgrande, J. P., Schaub, T., & Tompits, H. (2003). A framework for compiling preferences in logic programs. *Theory and Practice of Logic Programming*,*3*(02), 129-187.

Dhanesha, K. (1994). Normative expert system using deontic logic and defeasible reasoning. *Master's thesis*, The University of Georgia

Evans, R. (2009) Modeling Individual Personalities in The Sims 3. Game Developer's Conference, San Francisco, CA.

Evans, R. (2010) Sim Tribe: Using a Deontic Logic to Model Social Practices, in Experimental Game AI Live Demos, AI Summit Game Developer's Conference, San Francisco, CA.

Evans, R., & Short, E. (2014). Versu - A Simulationist Storytelling System. IEEE Transactions on Computational Intelligence and AI in Games, 6(2), 113–130.

Ghallab, M., Nau, D., & Traverso, P. (2004). Automated Planning: Theory and Practice. *Morgan Kaufmann Series in Artificial Intelligence*. Elsevier.

Horswill, I. D. (2014). Architectural Issues for Compositional Dialog in Games. In *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference.* May 2014, Raleigh, NC

Koons, R. (2014). "Defeasible Reasoning", *The Stanford Encyclopedia of Philosophy*, Edward N. Zalta (ed.), URL = http://plato.stanford.edu/archives/spr2014/entries/reasoning-defeasible.

Laird, J., Rosenbloom, P., and Newell, A. (1987). Soar: An Architecture for General Intelligence. Artificial Intelligence, 33: 1-64.

Mateas, M. & Stern, A. (2003). Façade: An Experiment in Building a Fully-Realized Interactive Drama. *In Game Developers Conference*, Vol 2

McCarthy, J. M. (1980) "Circumscription — A Form of Non-Monotonic Reasoning", *Artificial Intelligence*, 13: 27–39, 171–177.

McDermott, D. (1978), Planning and Acting. *Cognitive Science*, 2: 71–100.

Moore, R. C. (1985) "Semantic Considerations on Nonmonotonic Logic", *Artificial Intelligence*, 25: 75–94.

Nau, D., Cao, Y., Lotem,, A., & Muñoz-Avila, H. (1999). SHOP: Simple Hierarchical Ordered Planner. In *IJCAI-99*, pp. 968-973

Nute, D. (1993) Defeasible Prolog. Technical Report AI-1993-04, AI Programs, University of Georgia. Presented at AAAI Fall Symposium on Automated Deduction in Nonstandard Logics, Raleigh, NC

Pereira, F.C.N., & Shieber, S. (1987) *Prolog and Natural language Analysis*. Brookline, MA: Microtome Publishing.

Pollock, J. (1987). Defeasible Reasoning. *Cognitive Science,* 11, pp. 481-518

Pollock, J., (1992) How To Reason Defeasibly. *Artificial Intelligence* 57, pp 1-42

Pollock, J., (1998) Perceiving and Reasoning about a Changing World. *Computational Intelligence*, 14(4), 498-562

Pollock, J. (2000). Rational Cognition in OSCAR. Intelligent Agents VI. Agent Theories, Architectures, and Languages. In *Lecture Notes in Computer Science*, vol 1757, pp 71-90

Pollock, J., (2001) Defeasible Reasoning with Variable Degrees of Justification. *Artificial Intelligence* 133, pp 233-282

Reiter, R. (1980) "A logic for default reasoning", *Artificial Intelligence*, 13: 81–137.

Ryu, Y. (1992). A Formal Representation of Normative Systems: A Defeasible Deontic Reasoning Approach. *PhD thesis*, University of Texas.

Sergot, M. (2008). Action and agency in norm-governed multi-agent systems. In *Engineering Societies in the Agents World VIII* (pp. 1-54). Springer Berlin Heidelberg.

Sibun, P. (1992). *Locally Organized Text Generation*. University of Massachusetts, Amherst

Sterling, L. and Shapiro, E. (1994). *The Art of Prolog: Advanced Programming Techniques*. 2nd edition. MIT Press. Cambridge, MA.

Warren, D. H., & Pereira, F. C. (1982). An efficient easily adaptable system for interpreting natural language queries. *Computational Linguistics*, *8*(3-4), 110-122.

Zubek, R. 2010. "Needs-based AI." In A. Lake (ed.), *Game Programming Gems 8*, Cengage Learning, Florence, KY.