Revising Conflicting Intention Sets in BDI Agents*

Steven Shapiro
RMIT University
Melbourne, Australia
steven.shapiro@rmit.edu.au

Sebastian Sardina
RMIT University
Melbourne, Australia
sebastian.sardina@rmit.edu.au

John Thangarajah
RMIT University
Melbourne, Australia
john.thangarajah@rmit.edu.au

Lawrence Cavedon
RMIT University
Melbourne, Australia
lawrence.cavedon@rmit.edu.au

Lin Padgham
RMIT University
Melbourne, Australia
lin.padgham@rmit.edu.au

ABSTRACT

Autonomous agents typically have several goals they are pursuing simultaneously. Even if the goals themselves are not necessarily inconsistent, choices made about how to pursue each of these goals may well result in a set of intentions which are conflicting. A rational autonomous agent should be able to reason about and modify its set of intentions to take account of such issues. This paper presents the semantics of some preferences regarding modified sets of intentions. We look at the possibility of simply deleting some intention(s) but more importantly we also look at the possibility of modifying intentions, such that the goals will still be achieved but in a different way.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent Agents*

General Terms

Theory

Keywords

BDI, Logic-based approaches and methods, Formal models of agency, Modeling the dynamics of MAS

1. INTRODUCTION

BDI (Belief, Desire, Intention) systems (for an overview, see Bordini *et al.* [1]) are a popular approach to modelling and implementing agent systems. It is well accepted and reflected in the theoretical underpinnings of BDI systems (e.g., Rao and Georgeff [14]) that rational agents should not intend to pursue a goal that they believe is impossible to achieve. Similarly, we believe that an agent should not have a *set* of intentions, which taken together are unachievable. Such situations can readily arise as new goals are committed to, with new intentions instantiated, but without reasoning about how the independent intentions interact. Intention reconsideration—i.e., revisiting the commitments to planned activity held by an agent—is considered an important notion in BDI agent conceptual frameworks (e.g., Bratman [2] and Bratman *et al.* [3]). Intention revision is a central component of this process.

In this paper, we explore the semantics of the kinds of preferences that should guide the reasoning of an autonomous agent that is revising its intentions for some reason. Our focus is to provide a principled semantics for evaluating the different options for revising an existing intention set. The most straightforward way to revise an intention set is to drop one or more intentions. However, this will lead to lack of achievement of the associated goal(s). A preferred but more complex option is to modify one or more intentions (i.e., modify how we intend to achieve the associated goals) to obtain a non-conflicting intention set.

As an example, consider a purchasing agent with an intention to purchase a laptop and another intention to purchase a printer. If the agent is made aware that funds are insufficient for both purchases, then it may drop one intention such that the other is achievable. However, if it is possible to modify its current intentions, for example by changing its choice of laptop to be a cheaper model so that both intentions can be satisfied, this would be a preferred revision.

We consider three basic principles in defining our semantics: the first is *environmental tolerance*—we prefer a set of intentions which can succeed in more environments; the second is a *maximal cardinality* principle—keep as many top-level goals as possible; and the third is a *minimal modification* principle—if we must change the means chosen to achieve a goal (i.e., an intention), we prefer to change it as little as possible. These principles necessarily interact and we have to make particular choices as to which should dominate as we develop the semantics. However, minor modifications would allow the relationships to be changed.

Our long term goal is to incorporate rules into the execution engine of BDI agent programming languages to support principled intention revision. In doing this, we must choose between a quantitative framework—requiring costs associated with actions and rewards with goals, as well as perhaps probability distributions—and a qualitative one. The former would offer more flexibility, but requires the programmer to provide required quantities and measures. As typical BDI programming languages generally lack facilities for specifying such things as costs, rewards and probability distributions, we choose the qualitative approach. Consequently, we base our framework on the CAN family of languages [17] as this maps well to languages used in a number of BDI agent development platforms [1], but also has been extended to include advanced reasoning techniques, such as planning and reasoning about goals.

We stress that we are presenting here a semantics for how agents ought to revise their intentions, and not an implementation. However, in future work, we plan to develop an implementation that is both faithful to the semantics and computationally viable.

*We acknowledge the ARC Discovery grant DP1094627.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

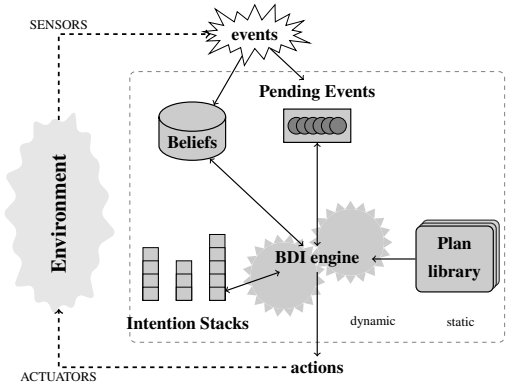

Figure 1: A typical BDI-style architecture [17].

In the following sections, we first present the basic constructs of the CAN language. We then provide the semantics for revising intention sets using only deletion, followed by the semantics when we allow modification of individual intentions. We finish with a discussion of related work, as well as some further issues to be addressed in the future.

2. AGENT PROGRAMMING IN THE CAN FRAMEWORK

BDI agent-oriented programming is based on formal computational models, such as the ones proposed by Cohen and Levesque [4] and Rao and Georgeff [14], and the philosophical work of Bratman [2] and Dennett [6], using mental attitudes, such as beliefs, desires, goals, plans and intentions. Practically, BDI agent systems enable *abstract plans* written by programmers to be combined and used in real-time, in a way that is both flexible and robust. The CAN (**C**onceptual **A**gent **N**otation) family of BDI languages [17] are AGENTSPEAK-like languages with a semantics capturing typical BDI systems (for an overview, see Bordini *et al.* [1]).

A typical BDI system (see Figure 1) responds to *events*[1] by selecting a plan from the system's know-how information for execution. The execution of a plan may, in turn, post new subgoal events to be achieved.

In the CAN language, there are three types of atoms: events (denoted by e), basic beliefs (denoted by b), and actions (denoted by *act*). A propositional language $\mathcal{L}_B$ is formed in the usual manner using the basic beliefs as atoms. We will use ψ, possibly with decorations, to denote a sentence of $\mathcal{L}_B$.

In CAN, a BDI agent is specified by an initial belief base $\mathcal{B}$, a plan library Π, and an action description library Λ. $\mathcal{B}$ is a model of the agent's initial beliefs about the world. It is a set of belief atoms that the agent holds to be true. We use $\mathcal{B} \models \psi$ to denote that the sentence ψ is true in belief base $\mathcal{B}$. This is defined in the usual manner.

An action description library Λ encodes the effects of primitive actions. It is a set of STRIPS-style operators of the form: $act : \psi \leftarrow \Phi^+; \Phi^-$, one for each action atom in the domain. Here, ψ is the precondition of the action, and is restricted to be a conjunction of belief literals. Φ^+ and Φ^- are sets of belief atoms, and correspond to STRIPS-style add and delete lists, respectively. Note that we assume all actions are deterministic.

[1]In CAN, there is no distinction between events and goals, and we sometimes refer to them as event-goals.

2.1 Plan Library

The *plan library*, Π, encodes the operational information of the domain via plan rules of the form $e : \psi \leftarrow P$, where e is an event, ψ is the context condition, and P is the plan-body program—*P is a reasonable strategy for resolving event e when condition ψ is believed to be true*. Plan-body programs are built from the following constructs, which we call the *user program language*:[2]

act	primitive action
$+b$, $-b$	add/delete a belief atom
$?\psi$	test for a condition
$!e$	post an event-goal
$P_1; P_2$	sequence

A program that only contains constructs from the user program language is called a *user program*.

In the *full program language*, there are two additional constructs used internally for defining the semantics of programs (i.e., they are not used in the programs in the plan library):

nil	the empty (terminating) program
$P \triangleright e : \langle\!\langle \psi_1 : P_1, \ldots, \psi_n : P_n \rangle\!\rangle$	attempt P to achieve e

In the last construct, P is a program, e is an event-goal, and:

$$\langle\!\langle \psi_1 : P_1, \ldots, \psi_n : P_n \rangle\!\rangle$$

is a set of alternative guarded plans relevant to e. The semantics for this construct is that an execution for P is attempted, and only if P fails, an alternate plan whose context condition is satisfied, say P_1, is selected for execution. In that case, the construct may transition to: $P_1 \triangleright e : \langle\!\langle \psi_2 : P_2, \ldots, \psi_n : P_n \rangle\!\rangle$.[3]

For example:[4]

$$!\mathit{Buy}(\mathit{pc}, \mathit{shop}) \triangleright \mathit{Get}(\mathit{pc}) : \langle\!\langle \mathit{isOnline} : !\mathit{Buy}(\mathit{pc}, \mathit{web}), \mathit{isOffline} : !\mathit{Buy}(\mathit{pc}, \mathit{mailOrder}) \rangle\!\rangle$$

could be the intention of a purchasing agent to attempt to purchase a laptop from the shop, and if that fails to fall back to achieving the goal $\mathit{Get}(\mathit{pc})$ either via the Web, if the agent is online, or by mail-order, if the agent is offline. There are standard ways to axiomatise a programming language such as the one presented here (see, e.g, Hennessy [10]), and we assume such an axiomatisation without presenting the details here. The language we use for this axiomatisation and for expressing properties of our framework is second-order, since we will be quantifying over functions, e.g., environments (which are defined below). We also assume we have an axiomatisation of finite sets, and omit the details here.

2.2 Intention Base

The *intention base*, Γ, of an agent contains the programs that the agent has already committed to for handling previously posted events. Formally, it is a set of programs in the full program language.[5] It is an element of the semantics of the CAN language, described below, but we present it separately since it is central to the framework described here; it is the intention base of an agent that is revised. We illustrate the intention base with an example.

[2]Note that the CAN language also contains a concurrency construct and allows variables in context conditions, but we omit those here for simplicity.

[3]Note that this is a variant of the CAN language, where $\triangleright$ and $e : \langle\!\langle \Delta \rangle\!\rangle$ are treated as separate constructs.

[4]As noted above, the version of CAN we consider here is propositional. However, we will sometimes use atoms with parentheses as syntactic sugar, e.g., $\mathit{Buy}(\mathit{pc}, \mathit{shop})$.

[5]In CAN, an intention is actually a pair containing an identifier (e.g., a natural number) and a program, however we suppress the identifier for simplicity.

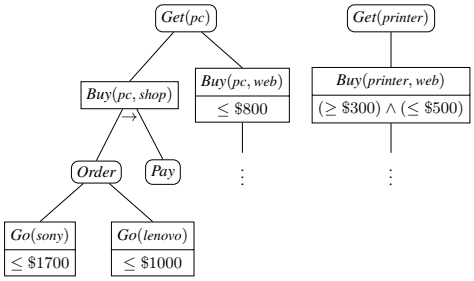

Figure 2: An example goal-plan hierarchy. Plans are represented with rectangles; goals are represented with rounded rectangles. Arcs with an arrow between them represent sequential composition. Arcs with no arrow represent alternate plans for their parent goal. The second annotation on plans states how much it will end up costing given the set of possible environments.

EXAMPLE 1. *Consider a purchasing agent with two intentions: one to achieve the goal of purchasing a laptop and another to achieve the goal of buying a printer. Figure 2 outlines the available plans in the library for achieving these goals. Assume* $\Gamma = \{I_1, I_2\}$*, where:*

$$I_1 = [(\mathit{Go}(\mathit{sony}) \rhd \mathit{Order} : (\!|\psi_1 : \mathit{Go}(\mathit{lenovo})|\!)); !\mathit{Pay}] \rhd \mathit{Get}(\mathit{pc}) : (\!|\psi_2 : \mathit{Buy}(\mathit{pc}, \mathit{web})|\!);$$
$$I_2 = !\mathit{Get}(\mathit{printer}).$$

That is, the agent has already started addressing the task of buying a laptop and has chosen to buy a Sony. However, should it fail to buy a Sony, it may alternatively try to buy a Lenovo (if context condition ψ_1 holds, e.g., the Lenovo store is open) or even fall back to acquiring a laptop on the Web (if condition ψ_2 applies, e.g., it can access the Internet).

2.3 Semantics

As with most agent programming languages, the Plotkin-style operational semantics of CAN closely follows Rao and Georgeff's [15] abstract interpreter for rational agents, and is defined using a so-called BDI *agent configuration* C of the form $\langle \Pi, \Lambda, \mathcal{B}, \mathcal{A}, \Gamma \rangle$, where components $\Pi, \Lambda, \mathcal{B}$, and Γ (resp.) are the plan library, action library, belief base and intention base (resp.), as described above, and $\mathcal{A}$ is the sequence of actions executed so far. A *transition relation* $C \longrightarrow_{\mathcal{E}} C'$ on agent configurations is then used to state that a *single step* in executing agent configuration C yields configuration C' within an environment $\mathcal{E}$. We represent an environment as a function $\mathcal{E} : \mathit{Confs} \mapsto 2^{\mathit{Events}}$, where *Confs* is the set of all possible agent configurations and *Events* is the set of all possible external events in the domain, i.e., programs of the form: $+b$, $-b$, and $!e$. That is, an environment determines which external actions occur at each step in the execution of an agent system. We further require that environments are consistent in the sense that at most one of $+b$ and $-b$ are in $\mathcal{E}(C)$, for any belief atom b and configuration C. For example, given a configuration C and an environment $\mathcal{E}$, $\mathcal{E}(C)$ might contain a belief update that the price of Lenovo laptops have dropped 30% (e.g., due to a sale). Finally, a *BDI execution* E of an agent $C_0 = \langle \Pi, \Lambda_0, \mathcal{B}_0, \mathcal{A}_0, \Gamma_0 \rangle$ in an environment $\mathcal{E}$ is a, possibly infinite, sequence of agent configurations $C_0 \cdot C_1 \cdots$ such that $C_i \longrightarrow_{\mathcal{E}} C_{i+1}$, for all $i \geq 0$.

Given an intention $I \in \Gamma_0$ and an execution E (w.r.t. an environment $\mathcal{E}$), it is possible to determine whether the intention I has successfully executed (i.e., it has evolved to the empty program, *nil*), failed, or can continue to execute. In general, an intention fails if it becomes *blocked*, that is, it cannot execute further in the current configuration (for example, if an action does not meet its precondition or a test step is believed false), and it cannot be "recovered" via the default failure handling mechanism of re-trying different alternatives for active (sub)goals. We refer to Sardina and Padgham [17] for full details on the semantics of the language.

In this paper, we focus mainly on the last component of configuration $C = \langle \Pi, \Lambda, \mathcal{B}, \mathcal{A}, \Gamma \rangle$, namely, the intention base Γ. It will be notationally convenient to separate this component out from the rest of the configuration. We therefore define the *diminished configuration* C_- to be a configuration with the intention base argument omitted, i.e., $C_- = \langle \Pi, \Lambda, \mathcal{B}, \mathcal{A} \rangle$. Where confusion does not arise, we will simply refer to these as *configurations*. $\langle C_-, \Gamma \rangle$ will be used to denote the (full) configuration that has Γ as its last component, and whose other components are as in C_-.

3. REVISION BY DROPPING INTENTIONS

Our aim is to characterise which possible intention bases Γ^* qualify as *acceptable intention revisions* for a given intention base Γ that the agent deems to be in need of reconsideration. We specify a *semantics* for when a new intention base counts as an appropriate revision of the current intentions. In future work, we plan to investigate an implementation that could be used in actual BDI systems which would be faithful to this ideal. We are currently agnostic on when and why a given intention base ought to be revised. This could happen, for example, when a new belief or intention is added, or when the agent determines there may be a negative interaction or conflict between intentions. One way to achieve a similar result to revising intentions would be to simply execute the intentions and the ones that end up being blocked would be dropped automatically from the intention set. However, executing intentions could consume valuable resources that could be saved by revising the intention set at appropriate times. In this section, we focus on the revision of intention sets by simply *abandoning* one or more current intentions.

One of the dimensions we use for comparing intention sets consists of the environments in which the intention set can be successfully executed. However, which intention sets can be successfully executed in an environment depends on how smart the agent is about the choices it has to make during an execution, e.g., about which plans to select and intentions to execute at which times. A smart agent, e.g., one who can plan ahead to make the right decisions, might be able to execute reliably a given intention set. On the other hand, a dumb agent—e.g., one who does not plan ahead, but rather chooses an intention to execute according to a fixed rule such as round-robin—might not be able to execute successfully the same intention set reliably. So, the definition of a successful execution depends on the agent under consideration. In the interest of generality—so that our framework can apply to a variety of agents—we take the successful execution of a set of intentions Γ in a configuration C_- and an environment $\mathcal{E}$, $\mathit{Success}(\Gamma, C_-, \mathcal{E})$, to be a primitive of our framework, and we leave it undefined. We only make the assumption that the empty set of intentions always executes successfully, since there is nothing to execute.

AXIOM 1 (**SUCCESS**).

$$\forall C_-, \mathcal{E}.\, \mathit{Success}(\emptyset, C_-, \mathcal{E}).$$

Let S denote this axiom together with the axioms for finite sets.

For example, let us return to our purchasing agent described in Example 1. Suppose the agent's budget is $1700 and that the set of possible environments are those described in Figure 2 with the

second annotation in plans (e.g., buying a computer on the web will cost no more than \$800). In this case, the agent may be unable to *successfully* execute (both) its intentions I_1 and I_2 (in environments where buying a Sony and a printer costs over \$1700) and hence one purchase may eventually fail (e.g., at time of payment for the printer, if the computer is purchased for \$1500). The question then is: *can the agent revise* Γ *to a more "robust" set of intentions?*[6]

The various dimensions in qualitative frameworks, such as the one we have chosen, are often incomparable, and therefore, they cannot be readily combined. Consequently, decisions must be made as to which dimensions take precedence. In our framework, one dimension for comparison consists of the environments in which an intention set can be successfully executed (*environmental tolerance*), and another is the cardinality of the intention set.[7] In this paper, we give precedence to environmental tolerance. However, the framework could easily be adapted so that cardinality gets precedence. Having environmental tolerance dominate cardinality means the agent is *cautious* in that it would prefer to drop an intention to gain more certainty that the intention set will be successfully executed. Alternatively, giving precedence to cardinality would model a *bold* agent that believes things will "work out" (with respect to the environment), and prefers to maintain more of its intentions. Both can lead to counterintuitive results in particular examples, but this is inherent in qualitative frameworks that contain incomparable dimensions.

At a minimum, once the agent has decided to revise its intentions, the revised intention set should be executable in *some* environment. We prefer intention sets that are executable in more environments. Of these preferred sets, we choose the sets that have higher cardinality, i.e., retain more intentions from the original set.

We define the revision of an intention set in three stages. First, we define what it means for an intention base to be *maximal* in terms of environmental tolerance. Amongst the maximal bases, the revision *candidates* are those that have some chance of success. We then say that the revision sets are the largest candidate sets.

3.1 Definition of a Maximal Set

We say that a set of intentions Γ *dominates* another set Γ' in a configuration C_-, if the agent can successfully execute Γ in a superset of the environments in which it successfully executes Γ' in C_-. Formally:

DEFINITION 2 (**DOMINATES**).

$$\mathit{Dom}(\Gamma, \Gamma', C_-) \stackrel{\mathit{def}}{=} \forall \mathcal{E}.\, \mathit{Success}(\Gamma', C_-, \mathcal{E}) \supset \mathit{Success}(\Gamma, C_-, \mathcal{E}).$$

We then define the set of maximal options in terms of dominance. A set of intentions $\overline{\Gamma}$ is maximal with respect to a set Γ in configuration C_-, if $\overline{\Gamma}$ is a subset of Γ and it dominates any nonempty subset of Γ that dominates it. The empty intention base needs to be ruled out for comparison as it (trivially) dominates *every* intention set—the empty set of intentions always executes successfully. We want to allow the possibility of the agent abandoning all its intentions, if no subsets of its intentions have any chance of success, however, we also want this to be the only condition under which the agent drops all of its intentions.

[6]Note that in our example, the order of the selection of plans has already been fixed, and the order of execution of intentions is irrelevant, therefore the exact definition of *Success* is not important.

[7]Later on, we will introduce a minimal modification dimension.

DEFINITION 3 (**MAXIMAL**).

$$\mathit{Max}(\overline{\Gamma}, \Gamma, C_-) \stackrel{\mathit{def}}{=} \overline{\Gamma} \subseteq \Gamma \wedge [\forall \Gamma'.\Gamma' \subseteq \Gamma \wedge \Gamma' \neq \emptyset \supset (\mathit{Dom}(\Gamma', \overline{\Gamma}, C_-) \supset \mathit{Dom}(\overline{\Gamma}, \Gamma', C_-))].$$

Observe here that the empty set is indeed maximal, but other options can be maximal as well.

3.2 Definition of a Revision Set

For an intention base to be a revision candidate of Γ, it must be a maximal subset of Γ, but also *possible* in the sense that the intentions can be achieved in at least one environment. Note that the second constraint does not follow from maximality: if no nonempty subset of Γ succeeds, then every subset will be considered maximal. Formally, we define $\mathit{Cand}(\cdot,\cdot,\cdot)$ as follows:

DEFINITION 4 (**REVISION CANDIDATE**).

$$\mathit{Cand}(\overline{\Gamma}, \Gamma, C_-) \stackrel{\mathit{def}}{=} \mathit{Max}(\overline{\Gamma}, \Gamma, C_-) \wedge \mathit{Poss}(\overline{\Gamma}, C_-), \textit{where}$$
$$\mathit{Poss}(\Gamma, C_-) \stackrel{\mathit{def}}{=} \exists \mathcal{E}.\, \mathit{Success}(\Gamma, C_-, \mathcal{E}).$$

Finally, an intention revision Γ^* of an intention base Γ is defined to be a largest candidate set:

DEFINITION 5 (**REVISION**).

$$\textsc{Rev}(\Gamma^*, \Gamma, C_-) \stackrel{\mathit{def}}{=} \mathit{Cand}(\Gamma^*, \Gamma, C_-) \wedge \forall \Gamma'.\, \mathit{Cand}(\Gamma', \Gamma, C_-) \supset |\Gamma'| \leq |\Gamma^*|.$$

In Example 1, there are two possible revisions: $\Gamma_1 = \{I_1\}$, and $\Gamma_2 = \{I_2\}$, i.e., the agent drops either one of its intentions. These sets dominate $\Gamma = \{I_1, I_2\}$ because they terminate in strictly more environments than Γ does (i.e., all environments where Sony + printer cost $>$ \$1700).

3.3 Properties

We now turn to some properties of these definitions.[8] Firstly, a (possibly empty) revision set always exists:

PROPOSITION 6.

$$S \models \forall \Gamma, C_-.\exists \Gamma^*.\, \textsc{Rev}(\Gamma^*, \Gamma, C_-).$$

Secondly, consider the (optimal) case in which it is possible to select a (nonempty) *fully robust* set of intentions, i.e., a set of intentions that can be fully executed in *every* environment. In that case, any revision will be fully robust:

PROPOSITION 7.

$$S \models \forall \Gamma, C_-.\ (\exists \Gamma'.\Gamma' \subseteq \Gamma \wedge \Gamma' \neq \emptyset \wedge \mathit{SuccessAlways}(\Gamma', C_-)) \supset (\forall \Gamma^*.\, \textsc{Rev}(\Gamma^*, \Gamma, C_-) \supset \mathit{SuccessAlways}(\Gamma^*, C_-)),$$
$$\textit{where } \mathit{SuccessAlways}(\Gamma, C_-) \stackrel{\mathit{def}}{=} \forall \mathcal{E}.\, \mathit{Success}(\Gamma, C_-, \mathcal{E}).$$

The empty set of intentions can always be achieved in any configuration. However, we want the agent to drop all its intentions *only* if there is no other choice, i.e., when none of the other subsets of Γ have any chance of success. Indeed, we can show that the revision of a set Γ is the empty set *iff* no nonempty subset of Γ is possible.

PROPOSITION 8.

$$S \models \forall \Gamma, C_-.[\forall \Gamma'.\Gamma' \subseteq \Gamma \wedge \Gamma' \neq \emptyset \supset \neg\, \mathit{Poss}(\Gamma', C_-)] \equiv [\forall \Gamma^*.\, \textsc{Rev}(\Gamma^*, \Gamma, C_-) \supset \Gamma^* = \emptyset].$$

[8]All propositions in this paper were verified with the PVS Verification System [12].

This proposition shows that given a set Γ that has no chance of success, although we cannot guarantee the success of a revision of Γ, any non-empty revision of Γ will at least have some chance of success. Furthermore, a revision of Γ will only be empty if no nonempty subset of Γ has any chance of success.

We close by noting that with the method for revision considered here, the only choice is to drop current intentions altogether. In the context of BDI agent systems, where goals can be achieved in multiple ways, one can envision more sophisticated accounts of revision so as to avoid resorting to such drastic decisions. We consider one such account in the next section.

4. REVISION BY MODIFICATION

In the previous section, we considered resolving conflicting intentions by dropping some. However, BDI agents often have various alternative plans for achieving goals. Instead of dropping an intention altogether, another option is to see if adopting other alternatives resolves the conflict. This option, which we call *intention modification* is preferable because it can lead to the achievement of more of the agent's goals.

Consider again our purchasing agent from Example 1 with the two intentions to buy a computer and a printer. Instead of dropping one of its intentions, the agent can consider modifying its intention to buy a computer by changing the method of achieving the event-goal *Order* to the plan $Go(lenovo)$, or indeed by changing the means to achieve the top-level event-goal $Get(pc)$ to the plan $Buy(pc, web)$. Either of these changes will result in the successful execution of both intentions, as the total cost in any possible environment will be below the available funds of \$1700.

The definition of revision by modifying intentions is more involved than the definition of revision by dropping intentions. First, we must define what we mean by modifying an intention. We modify an intention by selecting an alternate plan to achieve the intention or a subintention. This only makes sense for what we call *active goals* of the intention, which are the (sub)goals of the intention for which a plan has already been selected to achieve the goal. If P' is a modification of P, then we call P' an *alternate* of P. Then, we define the *alternate subset* relation which holds between two intention sets Γ' and Γ, if every element of Γ' is an alternate of an element of Γ. With this definition in hand, we define a *cardinal revision* of an intention set Γ in a similar fashion to the definition of revision in previous section, except we replace subset with alternate subset. This gives us a largest, maximal set in the space of alternate subsets of Γ, rather than in the space of subsets of Γ, as before. We define what it means to be a "least modification" of an intention, and generalise that definition to hold over sets of intentions, which we call *setwise closeness*. Finally, we define a revision of an intention set Γ^* to be a cardinal revision of Γ^* that is maximal with respect to setwise closeness. Although the formal framework in this section is more complicated than the previous one, conceptually the differences are simple: 1) the candidate sets for a revision of Γ are taken from the set of alternate subsets of Γ rather than the set of subsets of Γ; and 2) we add an extra dimension to compare the revision candidates for Γ, namely, minimal modification of the elements of Γ.

4.1 Definition of Alternate Subset

To define revision by modification, we must first define what we mean by a modification. We take a modification to be a different choice of a plan to achieve the intention or a subgoal of the intention. Formally, we say that P' is an *alternate* of a program P relative to a belief base $\mathcal{B}$, $P \rightsquigarrow_{\mathcal{B}} P'$ if P' can be obtained from P by changing the way some (sub)goal in P is to be achieved. We define this as a set of recursive axioms, with three cases depending on the structure of P.[9]

AXIOM 2 (**ALTERNATE OF A PROGRAM**).

$$(P_1; P_2) \rightsquigarrow_{\mathcal{B}} P' \equiv \exists P^*.P_1 \rightsquigarrow_{\mathcal{B}} P^* \wedge P' = (P^*; P_2);$$

$$P_1 \rhd e:(\!|\Delta|\!) \rightsquigarrow_{\mathcal{B}} P' \equiv (\exists P^*.P_1 \rightsquigarrow_{\mathcal{B}} P^* \wedge P' = (P^* \rhd e:(\!|\Delta|\!))) \vee \exists \psi, P.\ \psi:P \in \Delta \wedge \mathcal{B} \models \psi \wedge P' = (P \rhd e:(\!|\Delta \setminus \{\psi:P\}|\!));$$

$$P_1 \rightsquigarrow_{\mathcal{B}} P_2 \equiv P_1 = P_2, \textit{otherwise.}$$

In other words, if the program is a sequence $P_1; P_2$, then we recursively look for alternates in P_1. We do not need to consider alternates in P_2, since while the agent may have started executing P_1, and therefore may have generated alternative plans to achieve its (sub)goals, this would not yet be the case for P_2. If the program is of the form $P_1 \rhd e:(\!|\Delta|\!)$, then we recursively look for alternates in P_1 (first disjunct), but we also allow other choices of programs in Δ to achieve e, provided their context conditions are satisfied by $\mathcal{B}$ (second disjunct). In this case, P_1 is dropped as a means to achieve e. Note that when P_1 is dropped, *all* subgoals of P_1 are automatically dropped as well. For all other program constructs, the only alternate is the program itself.

EXAMPLE 9. *In Example 1, the alternates to I_1 are I_1 itself ($\rightsquigarrow_{\mathcal{B}}$ is reflexive), and assuming ψ_1 and ψ_2 are satisfied:*

$$I_1' = \big((Go(lenovo) \rhd Order:(\!|\,|\!)); !Pay\big) \rhd Get(pc):(\!|\psi_2 : Buy(pc, web)|\!);$$
$$I_1'' = Buy(pc, web) \rhd Get(pc):(\!|\,|\!).$$

The basic idea for revision by modification is that, given a set of intentions Γ to revise, we consider not just subsets of Γ, but also for each subset of Γ, we replace the elements of the subset by each of their alternates and consider the resulting sets as well. From these sets, we choose the ones with greatest environmental tolerance that are possible, and among those, the ones that are largest in size. From the remaining sets, we take the ones with the *least changes* to the elements of Γ to be the revisions of Γ. The aim in minimising the changes to the intentions is, as far as possible, to retain the work already done by the agent to achieve the associated goal.

Given the definition of an alternate of a program, we can now define the *alternate subset* relation ($\Gamma' \sqsubseteq_{\mathcal{B}} \Gamma$), which holds if every element of Γ' is an alternate of an element of Γ, and no two elements of Γ' are alternates of the same element of Γ:

DEFINITION 10 (**ALTERNATE SUBSET**).

$$\Gamma' \sqsubseteq_{\mathcal{B}} \Gamma \stackrel{def}{=} (\forall I' \in \Gamma'.\exists I \in \Gamma.I \rightsquigarrow_{\mathcal{B}} I') \wedge \forall I_1', I_2' \in \Gamma'.\forall I \in \Gamma.I \rightsquigarrow_{\mathcal{B}} I_1' \wedge I \rightsquigarrow_{\mathcal{B}} I_2' \supset I_1' = I_2'.$$

4.2 Definition of Cardinal Revision

In considering intention modifications, we only consider alternatives of *active goals* in each intention structure.[10] An active goal is the event-goal e in a program of the form $P \rhd e : (\!|\Delta|\!)$, where e is the active goal and Δ is a set of guarded plans.[11]

The *goal-program trace* of a program P is a finite sequence of pairs of the form $(e, (\!|\Delta|\!))$, consisting of an active goal e, and its alternate guarded plans $(\!|\Delta|\!)$, and is defined as follows. We treat this recursive definition as an axiom:

[9]We adopt the convention that unbound variables are universally quantified in the widest scope.

[10]All other goals are either completed, or not yet expanded so the alternatives are unknown.

[11]Note that this definition is different from the one given in [17], where the whole program was considered the active goal.

AXIOM 3 (**GOAL-PROGRAM TRACE**).

$$\mathit{Trace}(P) = \begin{cases} \mathit{Trace}(P_1) & \text{if } P = P_1; P_2 \\ (e, \langle\!|\Delta|\!\rangle) \cdot \mathit{Trace}(P_1) & \text{if } P = P_1 \rhd e : \langle\!|\Delta|\!\rangle \\ \epsilon & \text{otherwise,} \end{cases}$$

where the operator $\cdot$ is sequence concatenation and ϵ is the empty sequence. Note that we assume P has been evolved from a user program using the transition rules for CAN. For such programs, it can be shown that if P is of the form $P_1; P_2$ (case 1), none of the goals in P_2 are active yet. Therefore, we only have to consider the goal-program trace of P_1. For example, the goal-program trace of intention I_1 in Example 1 is:

$$(\mathit{Get}(pc), \langle\!|\psi_2 : \mathit{Buy}(pc, web)|\!\rangle) \cdot (\mathit{Order}, \langle\!|\psi_1 : \mathit{Go}(lenovo)|\!\rangle).$$

DEFINITION 11. *Let Σ denote the set of axioms consisting of Axioms 1–3, along with the axioms defining the language of* CAN*, and the axiomatisation of finite sets.*

Note that the framework in the previous section was independent of the agent language used, therefore we did not need the axioms for the language of CAN. However, revision by modification is framed in terms of the constructs of the CAN language. This is evident, for example, in Axiom 2. However, it would not be difficult to adapt the definitions to other agent languages.

We assume that each intention in the intention base has a different top-level goal. For $\Gamma \sqsubseteq_{\mathcal{B}} \Gamma^*$, this allows us to uniquely identify the subset of Γ^* whose alternate is Γ. The semantics of the CAN language ensures that an intention I in the intention base can only be of the form $!e$, or $P \rhd e : \langle\!|\Delta|\!\rangle$. We formally define the *top-level goal* of an intention as follows:

DEFINITION 12 (**TOP-LEVEL GOAL**).

$$\mathit{TLG}(I) \stackrel{def}{=} e, \textit{when } I = !e, \textit{or } \mathit{Trace}(I) = (e, \langle\!|\Delta|\!\rangle) \cdot \ldots$$

Note that the trace of $!e$ is ϵ, so the second case does not apply to intentions of the form $!e$. We say that a set of intentions is *distinct*, if the top-level goals of its members are all different:

DEFINITION 13 (**DISTINCT INTENTION SET**).

$$\mathit{Distinct}(\Gamma) \stackrel{def}{=} \forall I, I' \in \Gamma . I \neq I' \supset \mathit{TLG}(I) \neq \mathit{TLG}(I').$$

Next, we update the definitions *Max* and *Cand* from the previous section by substituting *subset* with *alternate subset* ($\sqsubseteq_{\mathcal{B}}$). Let C^B_- denote the belief base component of C_-. We define:

DEFINITION 14 (**MAXIMAL INTENTION SET**).

$$\mathit{Max}^+(\overline{\Gamma}, \Gamma, C_-) \stackrel{def}{=} \overline{\Gamma} \sqsubseteq_{C^B_-} \Gamma \wedge [\forall \Gamma' . \Gamma' \sqsubseteq_{C^B_-} \Gamma \wedge \Gamma' \neq \emptyset \supset (\mathit{Dom}(\Gamma', \overline{\Gamma}, C_-) \supset \mathit{Dom}(\overline{\Gamma}, \Gamma', C_-))].$$

DEFINITION 15 (**CANDIDATE INTENTION SET**).

$$\mathit{Cand}^+(\overline{\Gamma}, \Gamma, C_-) \stackrel{def}{=} \mathit{Max}^+(\overline{\Gamma}, \Gamma, C_-) \wedge \mathit{Poss}(\overline{\Gamma}, C_-).$$

In deciding which candidate sets are better than others, we now have two dimensions to consider. In the previous section, we chose the candidate sets with the largest cardinality. This amounts to preferring to keep top-level goals. As we will see below, for revision by modification, we will also prefer sets whose intentions have undergone the least degree of modification. For the moment, we use an intermediate predicate, REVCARD$^+$ to capture the former preference. We say that Γ^* is a *cardinal revision* of Γ in C_-, if Γ^* is a candidate for revision and is no smaller than all other candidates.

DEFINITION 16 (**CARDINAL REVISION**).

$$\text{REVCARD}^+(\Gamma^*, \Gamma, C_-) \stackrel{def}{=} \mathit{Cand}^+(\Gamma^*, \Gamma, C_-) \wedge \forall \Gamma' . \mathit{Cand}^+(\Gamma', \Gamma, C_-) \supset |\Gamma'| \leq |\Gamma^*|.$$

As one would expect, a cardinal revision always exists:

PROPOSITION 17.

$$\Sigma \models \forall \Gamma, C_- . \mathit{Distinct}(\Gamma) \supset \exists \Gamma^* . \text{REVCARD}^+(\Gamma^*, \Gamma, C_-).$$

4.3 Definition of Revision by Modification

Now, we formally specify what it means to be a "least modification" of an intention. Given belief base $\mathcal{B}$, and programs P, P_1 and P_2 such that $P \rightsquigarrow_{\mathcal{B}} P_1$ and $P \rightsquigarrow_{\mathcal{B}} P_2$, we say that P_1 is *closer* than P_2 to P (i.e., $P_1 \preceq_P P_2$), if the following holds (as standard, $|\cdot|$ denotes the length of a sequence):

DEFINITION 18 (**CLOSER**).

$$P_1 \preceq_P P_2 \stackrel{def}{=} |\mathit{Trace}(P_1)| \geq |\mathit{Trace}(P_2)| \wedge (P_2 = P \supset P_1 = P).$$

In other words, P_1 is closer to P than P_2 if, basically, no more goals have been dropped in P_1 than in P_2 (with respect to P). If no goals have been dropped in either, then P_2 is closer to P than P_1, if P_2 is identical to P, but P_1 is not (because in P_1, an alternate plan has been chosen to achieve the last subgoal). We capture that case via the second conjunct. For example, using the intentions in Examples 1 and 9, it can be seen that $I_1' \preceq_{I_1} I_1'' \wedge I_1'' \not\preceq_{I_1} I_1'$, since $|\mathit{Trace}(I_1')| > |\mathit{Trace}(I_1'')|$; I_1' is strictly closer to I_1 than I_1''. To verify that $\preceq_P$ is intuitively correct, we show that the length of the *Trace* of an alternate P' of a program P is not greater than the length of the *Trace* of P itself.

PROPOSITION 19.

$$\Sigma \models \forall \mathcal{B}, P, P' . P \rightsquigarrow_{\mathcal{B}} P' \supset |\mathit{Trace}(P)| \geq |\mathit{Trace}(P')|.$$

This property holds because an alternate P' of P can have dropped goals but not added any.

For our definition of revision by modification, we must generalise $\preceq_P$ to apply to two alternate subsets of the set Γ of intentions to be revised. We only compare sets that consist of alternates of the same subset of Γ. If $\Gamma' \sqsubseteq_{\mathcal{B}} \Gamma$, then the elements of Γ' are each alternates of the elements of a subset of Γ, which we call the *core* of Γ' with respect to Γ and $\mathcal{B}$. Formally, given $\Gamma' \sqsubseteq_{\mathcal{B}} \Gamma$:

DEFINITION 20 (**CORE OF AN INTENTION SET**).

$$\mathit{core}(\mathcal{B}, \Gamma, \Gamma') \stackrel{def}{=} \{P \in \Gamma | \exists P' \in \Gamma' . P \rightsquigarrow_{\mathcal{B}} P'\}.$$

Note that if Γ is distinct, then Γ' is guaranteed to be the same size as $\mathit{core}(\mathcal{B}, \Gamma, \Gamma')$, for any $\mathcal{B}$.

For alternate subsets Γ_1 and Γ_2 of Γ that share the same core, we say that Γ_1 is *setwise closer* to Γ than Γ_2, if for every $I \in \Gamma$, I's alternate in Γ_1 is closer to I than its alternate in Γ_2:

DEFINITION 21 (**SETWISE CLOSER**).

$\Gamma_1 \preceq_{\mathcal{B},\Gamma} \Gamma_2 \stackrel{def}{=}$
if $\Gamma_1 \sqsubseteq_{\mathcal{B}} \Gamma \wedge \Gamma_2 \sqsubseteq_{\mathcal{B}} \Gamma \wedge \mathit{core}(\mathcal{B}, \Gamma, \Gamma_1) = \mathit{core}(\mathcal{B}, \Gamma, \Gamma_2)$
then $\forall I \in \Gamma, I_1 \in \Gamma_1, I_2 \in \Gamma_2 . I \rightsquigarrow_{\mathcal{B}} I_1 \wedge I \rightsquigarrow_{\mathcal{B}} I_2 \supset I_1 \preceq_I I_2$
else $FALSE$,

where **if** A **then** B **else** $C \stackrel{def}{=} (A \supset B) \wedge (\neg A \supset C)$.

Finally, we say that Γ^* is a *modification revision* of Γ in C_-, if Γ^* is a cardinal revision and is maximal with respect to setwise closeness:

DEFINITION 22 (**MODIFICATION REVISION**).

$$\textsc{Rev}^+(\Gamma^*, \Gamma, C_-) \stackrel{def}{=} \textsc{RevCard}^+(\Gamma^*, \Gamma, C_-) \wedge \forall \Gamma'. \textsc{RevCard}^+(\Gamma', \Gamma, C_-) \supset (\Gamma' \preceq_{C_-^{\mathcal{B}}, \Gamma} \Gamma^* \supset \Gamma^* \preceq_{C_-^{\mathcal{B}}, \Gamma} \Gamma').$$

4.4 Properties

We can show that (the appropriate reformulations of) the propositions in the previous section hold for $\textsc{Rev}^+$ as well.

A (possibly empty) modification revision set of a distinct set Γ always exists.

PROPOSITION 23.

$$\Sigma \models \forall \Gamma, C_-. \mathit{Distinct}(\Gamma) \supset \exists \Gamma^*. \textsc{Rev}^+(\Gamma^*, \Gamma, C_-).$$

If there exists a fully robust subset of a distinct set Γ, then the modification revision set will be fully robust.

PROPOSITION 24.

$$\Sigma \models \forall \Gamma, C_-. \mathit{Distinct}(\Gamma) \wedge (\exists \Gamma'. \Gamma' \sqsubseteq_{C_-^{\mathcal{B}}} \Gamma \wedge \Gamma' \neq \emptyset \wedge \mathit{SuccessAlways}(\Gamma', C_-)) \supset (\forall \Gamma^*. \textsc{Rev}^+(\Gamma^*, \Gamma, C_-) \supset \mathit{SuccessAlways}(\Gamma^*, C_-)).$$

The modification revision of a distinct set Γ is the empty set *iff* no nonempty subset of Γ is possible.

PROPOSITION 25.

$$\Sigma \models \forall \Gamma, C_-. \mathit{Distinct}(\Gamma) \supset [\forall \Gamma'. \Gamma' \sqsubseteq_{C_-^{\mathcal{B}}} \overline{\Gamma} \wedge \Gamma' \neq \emptyset \supset \neg \mathit{Poss}(\Gamma', C_-)] \equiv [\forall \Gamma^*. \textsc{Rev}^+(\Gamma^*, \Gamma, C_-) \supset \Gamma^* = \emptyset].$$

Let us return to our purchasing agent example, as described in Examples 1 and 9, and in Figure 2. As there is \$1700 available, the (best) revision is obtained by dropping the goal of buying a Sony and adopting the goal of buying a Lenovo (for achieving event-goal *Order*), that is, the only modification revision is $\Gamma^* = \{I_1', I_2\}$. Observe that while intention base $\Gamma' = \{I_1'', I_2\}$ can be considered for revision, it involves higher-level modifications to intentions than Γ^* does. That is, let $\mathcal{B}$ be the agent's current belief base, then $I_1' \preceq_I I_1'' \wedge I_1'' \not\preceq_I I'$, and therefore $\Gamma^* \preceq_{\mathcal{B},\Gamma} \Gamma' \wedge \Gamma' \not\preceq_{\mathcal{B},\Gamma} \Gamma^*$, which means that Γ' does not qualify as an acceptable revision set. However, if the agent were to believe there was only \$1400 available, then intention I_1 would have to be modified higher up in its hierarchy of goals, and the only revision would indeed be Γ'. Observe that in both cases, though, dropping any of the intentions, as in the previous section, would not qualify as acceptable revisions: there is something less drastic that can be done.

5. RELATED WORK

The problem of revising intentions, especially within formal BDI frameworks, has received surprisingly little attention. Rao and Georgeff [16] extend their seminal BDI logic to resolve a specific paradox that arises when intentions are dropped but do not consider the general problem. Wobcke [19] considers the effects on intentions when beliefs are revised; he uses a version of *epistemic entrenchment* [7], effectively requiring a quantitative measure of the priority of each intention. To our knowledge, Wobcke was the first to apply ideas from belief revision [7] to the problem of revising intentions. However, his framework, and in particular his representation of plans, is restrictive.

More recently, Grant *et al.* [8] present a detailed investigation of the general problem of intention revision. They propose postulates for the revision of *BDI structures*; a BDI structure $\langle B, D, I, v, (c, C)\rangle$ is a representation of the current mental state of an agent (similar to what we call a configuration), where v computes the "value" in achieving a desire and c assigns a "cost" to performing actions in the domain C (which includes all actions the agent may intend) that are used to achieve goals. Each intention in I is represented as a plan instance or "recipe" $a \rightarrow g$, where a is an action to perform and g is the goal (i.e., proposition) that would be achieved by successful execution of a. Grant *et al.* are specifically interested in BDI structures that are "rational" (are not internally inconsistent) and that maximise "benefit" (the total value of its current goals minus the cost of the actions to achieve them).[12]

Grant *et al.* propose postulates for the various cases of adding and deleting a belief, desire or intention, and for the cases of modifying value and cost functions; each of these cases may result in a new BDI structure. Their main requirements for the resulting BDI structure are: (i) the rationality condition; (ii) that any change to the intention set is minimal (in that it overlaps as much as possible with the intention set in the original BDI structure); and (iii) that the candidate revisions are maximal in benefit.

While our aims have many similarities to those of Grant *et al.*, we have taken an orthogonal approach by considering intention revision within the rich CAN framework. In particular, this framework allows us to reason about complex plans or recipes to be used to achieve goals, and the environments in which they can be successfully performed, allowing us to revise to intention sets that are maximally likely to be successful. The rich plan representation, including subgoals, available in the CAN language also allows us to consider the possibility of revising to a new intention by considering alternative execution paths to achieve a goal or subgoal. In particular, use of the CAN framework better bridges the gap between abstract theoretical framework and the standard execution model of BDI agent systems [1, 13]. It also allows us to reason explicitly about revising to sets of maximally robust intentions, and impose our minimal modification condition.

Van der Hoek *et al.* [18] define a powerful framework for describing and reasoning about BDI mental states, and a corresponding account for revising intentions. Their ultimate goal is to incorporate the dynamics of intentions into a framework for reasoning about mental states; as such, they have similar aims to ours. Their formal framework is very rich; however, their approach to the actual intention revision is very algorithmic, which is an issue that Grant *et al.* [8] specifically attempt to address. Further, the model presented in Van der Hoek *et al.* does not address two of our main concerns: their intention-removal step does not seem to account for mutually conflicting intentions; and they do not enforce minimal modification of intentions.

Finally, Icard *et al.* [11] also consider how belief revision impacts intention revision; they also tightly intertwine the revision of beliefs and intention sets. They provide postulates for revising both sets of intentions and beliefs. However, this is under a simple model of plans, in particular, non-hierarchical plans; hence, they are not able to capture a notion of minimal plan modification such as the one presented here.

Most of the other frameworks discussed here examine the relationship between beliefs, goals and intentions, which is something

[12] In a follow-up paper, Grant *et al.* [9] extend this framework to the multi-agent case, to model a different problem, that of how agents use knowledge of each others' intentions to coordinate behaviour. Since this work is only tangentially related to ours, we will not discuss it further here.

which we do not address. The reason we do not address this issue is that although the interplay between these mental attitudes is theoretically interesting, we do not see it playing a role in currently implemented systems. For example, in the CAN framework, as well as in many implemented agent systems, goals are simply labels with no propositional meaning. Similarly, while beliefs do have propositional content in CAN, the language in which beliefs are framed does not admit beliefs about the future. Since goals and intentions are future-oriented, in the absence of beliefs about the future, the relationship between beliefs on the one hand, and goals and intentions on the other is not very interesting. There are implemented systems (e.g., [5]) that have goals with propositional content, however, as far as we are aware, there are no implemented systems that allow explicit beliefs about the future.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have addressed an important aspect of agent reasoning—intention revision. Intention revision is central, since an agent typically pursues multiple tasks, adopting intentions to achieve them. These intentions may conflict with each other, due to resource limitations, for example. In the event of such conflicts the agent ought to resolve them in a *rational* manner.

We have defined a framework for the revision of sets of intentions that mutually conflict. We have presented two approaches. The first is to revise the set of intentions by dropping one or more intentions to attain a non-conflicting set, and the other is to modify the current intentions so that they may be achieved by alternative means to obtain a non-conflicting set of intentions.

Our theory of intention revision is embedded in a powerful formal framework for the representation of goals, programs, and the environments under which they are executed. This framework allows us to specify rich criteria for appropriate revised sets of intentions to satisfy: *robustness* (guaranteed success in maximal number of environments), minimal reduction on dropping an intention (i.e., preserving maximal number of top-level intentions), and minimal change to the means of achieving an intention.

Future work includes extending our model to include further features of Grant *et al.*'s framework, i.e., adding new intentions, representing costs of action and value of achieving goals. In such an account, we would expect to be able to demonstrate satisfaction of their postulates within our much richer framework of goals and plans, providing a clear specification of intention revision behaviour for BDI agent programming models. Our ultimate aim is a model of the complete intention reconsideration problem, including intention selection, opportunistic merging, and revision; and its application to the design of such processes in practical agent programming frameworks. In this paper, we presented a semantics for intention revision and future work also involves developing an implementation that is faithful to these semantics but also computationally feasible.

7. REFERENCES

[1] R. H. Bordini, L. Braubach, M. Dastani, A. Fallah-Seghrouchni, J. J. Gómez Sanz, J. Leite, G. O'Hare, A. Pokahr, and A. Ricci. A survey of programming languages and platforms for multi-agent systems. *Informatica (Slovenia)*, 30(1):33–44, 2006.

[2] M. E. Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press, 1987.

[3] M. E. Bratman, D. J. Israel, and M. E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(3):349–355, 1988.

[4] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[5] F. S. de Boer, K. V. Hindriks, W. van der Hoek, and J.-J. Meyer. A verification framework for agent programming with declarative goals. *Journal of Applied Logic*, 5(2):277–302, 2007.

[6] D. Dennett. *The Intentional Stance*. MIT Press, 1987.

[7] P. Gärdenfors. *Knowledge in Flux*. The MIT Press, 1988.

[8] J. Grant, S. Kraus, D. Perlis, and M. Wooldridge. Postulates for revising BDI structures. *Synthese*, 175:127–150, 2010.

[9] J. Grant, S. Kraus, and M. Wooldridge. Intentions in equilibrium. In M. Fox and D. Poole, editors, *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 786–791. AAAI Press, 2010.

[10] M. Hennessy. *The Semantics of Programming Languages*. John Wiley & Sons, Chichester, England, 1990.

[11] T. Icard, E. Pacuit, and Y. Shoham. Joint revision of beliefs and intentions. In F. Lin, U. Sattler, and M. Truszczynski, editors, *Proceedings of Principles of Knowledge Representation and Reasoning (KR)*, pages 572–574. AAAI Press, 2010.

[12] S. Owre, S. Rajan, J. M. Rushby, N. Shankar, and M. K. Srivas. PVS: Combining specification, proof checking, and model checking. In R. Alur and T. A. Henzinger, editors, *Proceedings of the International Conference on Computer Aided Verification (CAV)*, volume 1102 of *Lecture Notes in Computer Science (LNCS)*, pages 411–414. Springer-Verlag, 1996.

[13] A. S. Rao. Agentspeak(L): BDI agents speak out in a logical computable language. In W. V. de Velde and J. W. Perram, editors, *Proceedings of the European Workshop on Modelling Autonomous Agents in a Multi Agent World (MAAMAW)*, volume 1038 of *Lecture Notes in Computer Science (LNCS)*, pages 42–55. Springer, 1996.

[14] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In J. F. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of Principles of Knowledge Representation and Reasoning (KR)*, pages 473–484. Morgan Kaufmann, 1991.

[15] A. S. Rao and M. P. Georgeff. An abstract architecture for rational agents. In B. Nebel, C. Rich, and W. R. Swartout, editors, *Proceedings of Principles of Knowledge Representation and Reasoning (KR)*, pages 438–449. Morgan Kaufmann, 1992.

[16] A. S. Rao and M. P. Georgeff. BDI agents: From theory to practice. In V. Lesser and L. Gasser, editors, *Procedings of the International Conference on Multiagent Systems (ICMAS)*, pages 312–319. AAAI Press / MIT Press, 1995.

[17] S. Sardina and L. Padgham. A BDI agent programming language with failure recovery, declarative goals, and planning. *Autonomous Agents and Multi-Agent Systems*, 23(1):18–70, 2011.

[18] W. van der Hoek, W. Jamroga, and M. Wooldridge. Towards a theory of intention revision. *Synthese*, 155(2):265–290, 2007.

[19] W. Wobcke. Plans and the revision of intentions. In C. Zhang and D. Lukose, editors, *Distributed Artificial Intelligence: Architecture and Modelling*, volume 1087 of *Lecture Notes in Computer Science (LNCS)*, pages 100–114. Springer, 1995.