Practical reasoning about complex activities

Esteban Guerrero and Helena Lindgren

Computing Science Department.
Umeå University. Umeå, Sweden,
(esteban.guerrero, helena.lindgren)@umu.se

Abstract. In this paper, we present an argument-based mechanism to generate hypotheses about belief-desire-intentions on dynamic and complex activities of a software agent. We propose to use a composed structure called *activity* as unit for agent deliberation analysis, maintaining actions, goals and observations of the world always situated into a context. Activity transformation produces changes in the knowledge base activity structure as well in the agent's mental states. For example, in car driving as a changing activity, experienced and novice drivers have a different mental attitudes defining distinct deliberation processes with the same observations of the world. Using a framework for understanding activities in social sciences, we endow a software agent with the ability of deliberate, drawing conclusion about current and past events dealing with activity transformations. An argument-based deliberation is proposed which progressively reason about activity segments in a bottom-up manner. Activities are captured as extended logic programs and hypotheses are built using an answer-set programming approach. We present algorithms and an early-stage implementation of our argument-based deliberation process.

Keywords: practical reasoning, agents, complex activity, argumentation, deliberation, tool

1 Introduction

In social sciences, an *activity*[1] in general is understood as a purposeful interaction of the subject with the world [15]. This activity-theoretical concept [20] has been used to frame human behavior around the conscious pursue of goals to fulfill human needs. Key element of this theory is the concept of activity as a complex, dynamic and hierarchical structure. Among other approaches from social sciences, activity theory (AT) has been typically used for describing and explaining past events, for instance investigating activity dynamics considering current situations.

On the other hand, in artificial intelligence, *practical reasoning* investigates about what it is best for a particular agent to do in a particular situation [3]. Roughly, it explores the pursuing of goals by rational agents through two processes: *deliberation* deciding which of a set of options an agent should pursue; and *means-end reasoning*, solving the question how to achieve the selected goal. In other words, this models endow agents with abilities to plan ahead.

[1] Not only human activity but activity of any subject.

This work addresses the research question: how a software agent can look ahead for the next goal to execute when current and past events are considered? This problem is solved in two phases: 1) framing the evaluation of current and past events under an *activity analysis*, using AT to structure the agent knowledge, and argumentation theory[2] to deliberate about it; and 2) planning ahead using consistent hypothesized intentions which follow a well-known approach in practical reasoning, the *Belief*, *Desire* and *Intention* (BDI) model [8, 25].

In our approach, deliberation is performed using a bottom-up method, drawing conclusions progressively using results from previous computation, *i.e.,* explanations in the *operative level* about atomic no-purposeful elements of an activity are generated; then explanations in the *objective level* about purposeful goals and conditions that need to be hold (from operative level) are build; then, explanations in the *intentional level* conclusions about an explicit conscious action to perform a goal under certain circumstances (from operative and objective level) are generated. In summary, the following technical contributions are presented: 1) a notion of practical reasoning about complex agent activities; 2) a progressive bottom-up deliberation based on answer-set programming and argumentation theory; 3) algorithms for practical reasoning; and 4) an open source tool for argument-based deliberation on complex activities.

The paper is organized as follows. In Section 2 we introduce basic notions about what a dynamic and complex activity is along with the syntax language that we use in the paper. In Section 3 we present our main contributions, where the deliberation process is formalized and exemplified. We implemented a first step on practical reasoning on activities developing a Java-based tool described in Section 4; in this section we also introduce algorithms that were implemented on the tool. In Section 5 we discuss about our approach regarding close related work. We highlight our main contributions in Section 6.

2 Preliminaries

2.1 Dynamic activities

Activity theory defines an *activity* as a hierarchical structure composed by *actions*, which are, in turn, composed of *operations*. These three levels correspond, respectively, to *motives*, *goals*, and *conditions*, as indicated by arrows in Figure 1. According to AT, actions are directed to goals; goals are conscious, *i.e.,* a human agent is aware of goals to attain. Actions, in their turn, can also be decomposed into lower-level units of activity called *operations*. Operations are routine processes providing an adjustment of an action to the ongoing situation, they are oriented toward the *conditions* under which the agent is trying to attain a goal. In this paper, an activity can be defined by the tuple $\mathcal{A} = \langle Go, Ac, Op, Co \rangle$ where $Go = \{g_1, \ldots, g_i\}$ is the set of $i > 0$ goals of the activity; $Ac = \{ac_1, \ldots, ac_j\}$ is the set of $j > 0$ actions associated with the set Go; $Op = \{op_1, \ldots, op_k\}$ is the set of $k > 0$ operations; and $Co = \{co_1, \ldots, co_l\}$ is the set of $l > 0$ conditions related to operations.

[2] A general perspective about argumentation theory is presented in [4]

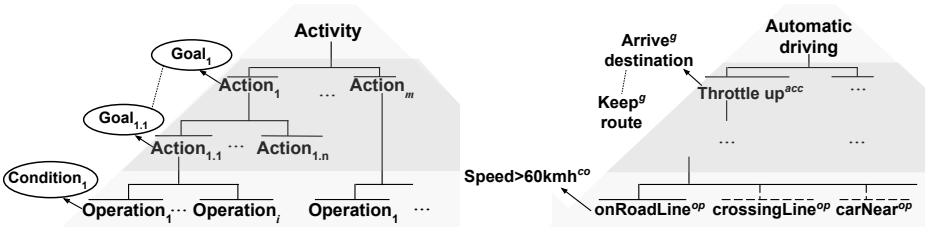

Fig. 1. The hierarchical structure of activity in activity theory. Adapted from [15]

2.2 Underlying logical language

In the hierarchical structure of an activity mentioned above, the agent current state depends on external information to the agent's knowledge base. This information can be incomplete or uncertain. In order to capture and deals with this information during the deliberation process we use logic programs with negations as failure (NAF).

We use a propositional logic with a syntax language constituted by propositional symbols: $p_0, p_1, \ldots$; connectives: $\wedge, \leftarrow, \neg,\ not, \top$; and auxiliary symbols: (,), in which $\wedge, \leftarrow$ are 2-place connectives, $\neg,\ not$ are 1-place connectives and $\top$ is a 0-place connective. Propositional symbol $\top$ and symbols of the form $\neg p_i (i \geqslant 0)$ stand for indecomposable propositions which we call *atoms*, or *atomic propositions*. Atoms of the form $\neg a$ are called *extended atoms* in the literature. An *extended normal clause*, C, is denoted:$a \leftarrow b_1, \ldots, b_j,\ not\ b_{j+1}, \ldots, not\ b_{j+n}$ where $j + n \geqslant 0$, a is an atom and each $b_i (1 \leqslant i \leqslant j + n)$ is an atom. When $j + n = 0$ the clause is an abbreviation of $a \leftarrow \top$ such that $\top$ always evaluates true. An *extended normal program* P is a finite set of extended normal clauses. By $\mathcal{L}_P$, we denote the set of atoms which appear in a program *P*. ELP use both strong negation $\neg$ and *not*, representing commonsense knowledge through logic programs. On programs with NAF, the consequence operator: $\leftarrow$ is not monotonic, which means that the evaluation result, may change as more information is added to the program. Two major semantics for ELP have been defined: 1) answer set semantics [11], an extension of *Stable model semantics*, and 2) the Well-Founded Semantics (WFS) [27]. Let $ASP(S)$ be a function returning a *semantic evaluation*[3] of a set $S \subseteq P$ in which any of these two ELP semantics is used. In consequence, the range of this function is: $ASP(S) = \langle T, F \rangle$. Roughly speaking, $ASP()$ will return true (T) or false (T) for a given set S. ASP function will be use to make a consistency checking of rules sets, dealing with possible inconsistencies of the agent's activity *e.g.,* detecting "loops" such as $S = \{a \leftarrow not\ b,\ b \leftarrow not\ a\}$.

In order to exemplify our approach, we introduce an example about how an activity can be captured using this underlying formalism:

[3] Semantic in terms of a semantic system [23]. A semantic system relates a set F of logical formulae to a set M of formal models, each representing a conceivable state of the world in enough detail to determine when a given formula represents a true assertion in that state of the world.

Example 1. A rational agent is deployed in a self-driving vehicle [4]. In this context, *driving* is an activity for the agent. This example is reduced to exemplify rational deliberation only. This activity consists of different actions, goals, operations and conditions which are indicated by superscripts ^{acc},^{g} ,^{op} and ^{co} respectively as follows:

$$P := \left\{ \begin{array}{c} \neg legalSpeed^{g} \leftarrow speed > 60kmh^{co} \wedge limitSign^{op} \wedge idle^{acc} \\ avoidCollision^{g} \leftarrow carNear^{op} \wedge carDist < 10m^{co} \wedge steeringLeft^{acc} \\ arriveDestination^{g} \leftarrow keepRoute^{g} \ \wedge \ throttleUp^{acc} \\ keepRoute^{g} \leftarrow onRoadLine^{op} \wedge not\ carNear^{op} \wedge speed > 60kmh^{co} \\ \ldots \\ verifySocialNet^{g} \leftarrow touchScreen^{op} \ \wedge internetAvailab^{co} \\ \ldots \\ inMove^{op} \leftarrow onRoadLine^{op} \wedge speed > 0kmh^{co} \\ carNear^{op} \leftarrow not\ carDistant^{op} \\ safeSpeed^{op} \leftarrow not\ limitSign^{op} \\ onRoadLine^{op} \leftarrow not\ crossingLine^{op} \\ crossingLine^{op} \leftarrow not\ onRoadLine^{op} \\ throttleDown^{acc} \leftarrow not\ throttleUp^{acc} \\ throttleUp^{acc} \leftarrow not\ throttleDown^{acc} \\ \ldots \end{array} \right\}$$

$$G := \{legalSpeed^{g},\ \neg legalSpeed^{g},\ avoidCollision^{g},\ keepRoute^{g}\}$$

$$Ac := \left\{ \begin{array}{c} brakeDown^{acc},\ idle^{acc},\ steeringLeft^{acc}, \\ throttleDown^{acc},\ throttleUp^{acc} \end{array} \right\}$$

$$Op := \left\{ \begin{array}{c} limitSign^{op},\ carNear^{op},\ onRoadLine^{op}, \\ crossingLine^{op},\ safeSpeed^{op},\ carDistant^{op},\ carNear^{op} \end{array} \right\}$$

$$Co := \left\{ \begin{array}{c} morning^{co},\ evening^{co}, \\ speed > 60kmh^{co},\ carDist < 10m^{co} \end{array} \right\}$$

In P, an intuitive reading of a clause *e.g.*: $keepRoute^{g} \leftarrow onRoadLine^{op} \wedge not\ carNear^{op}$, indicates that given that there is not evidence about a car nearby and the vehicle is in the road line, then the vehicle keeps its route.

Relevance is a property that some logic programming semantics satisfies, including WFS. The *relevant rules* of a program P *w.r.t.* a literal L contains all rules, that could ever contribute to L's derivation. Roughly speaking, the truth-value of an atom, *w.r.t.* any semantics, only depends on the *subprograms* formed from the *relevant clauses* with respect to that specific atom [7].

Definition 1. *Let P be an extended logic program capturing an activity $\mathcal{A}$ and let $x \in \mathcal{L}_P$ be an action or operation in $\mathcal{A}$.* $\mathsf{rel\_rules}(P, x)$ *is a function which returns the set of clauses containing $a \in \mathsf{dependencies\_of}(x)$ in their heads.*

Example 2. Following Example 1, we can obtain related rules from a given action, *e.g.,* $steeringLeft^{acc}$ as follows: $\mathsf{rel\_rules}(P, steeringLeft^{acc}) = \{avoidCollision^{g} \leftarrow carNear^{op} \wedge carDist < 10m^{co} \wedge steeringLeft^{acc}\}$

[4] Some actions and operations are based on a self-driving vehicle example in [22]

3 Deliberation on activities

Deliberation is performed on related information about an activity *w.r.t.* a particular atom, *e.g.* an operation or an action. Our bottom-up approach for deliberation starts with an analysis in the operative level of the activity as follows.

3.1 Deliberation in the operative level

According to AT, an activity analysis in the operative level implies the examination of processes that become a routine [15]. For a rational agent, the importance of building operative level hypotheses lies in dealing with uncertainty of the external world observations, handling inconsistencies of its internal knowledge base and reasoning about *belief* routines. Hypotheses at this level can be built as follows:

Definition 2 (Operative hypothesis). *Let $\mathcal{A} = \langle Go, Ac, Op, Co \rangle$ be an agent activity. Let $S \subseteq P$ be a subset of an extended logic program; let $op \in Op$ be an operation and let $R = \mathsf{rel\_rules}(S, op)$ be the set of clauses related to op. An operative level hypothesis is a tuple $H_{op} = \langle R, op \rangle$ if the following conditions hold:*

1. *$ASP(R) = \langle T, F \rangle$ such that $op \in T$.*
2. *R is minimal w.r.t. the set inclusion, satisfying condition 1.*
3. *$\nexists\ op \in \mathcal{L}_P$ such that $\{op, \neg op\} \subseteq T$ and $ASP(R) = \langle T, F \rangle$.*

where $Op, Co \subseteq R$.

An operative hypothesis as is presented in Definition 2, defines a consistent knowledge structure allowing to an agent ascertain about a reliable belief about the world. Moreover, the first step in Definition 2 can be seen as a consistency checking process for dealing with uncertain information of the current belief.

Example 3. Let us continue with Example 1. Using P the following is an operative hypothesis that an agent can build from its driving activity:

$$H_{op_1} = \langle \underbrace{\{inMove^{op} \leftarrow onRoadLine^{op} \wedge ...; onRoadLine^{op} \leftarrow not\ crossLine^{op}\}}_{S}, \underbrace{inMove^{op}}_{op} \rangle$$

H_{op_1} says that there is consistent and well-supported evidence that the vehicle is in movement $inMove^{op}$[5].

Operations in AT are well-defined routines [18], *e.g.* in driving, as an agent's activity, the continuous verification to keep the vehicle on the road line can be considered as an operation, a routine. In this context, a sub-routine example can be collect information about distance between the road line and the vehicle wheel location. Sub-routines can be also captured using the concept of sub-operative hypotheses as follows:

Definition 3. *Let $H_{op_A} = \langle R_A, op_A \rangle$, $H_{op_B} = \langle R_B, op_B \rangle$ be two operative hypotheses. H_{op_A} is a sub-operative hypotheses of H_{op_B} if and only if $R_A \subset R_B$.*

In Example 3, a sub-operative hypothesis can also be built from the *atomic* rule: $onRoadLine^{op} \leftarrow not\ crossLine^{op}$, *e.g.*:

$$H_{sub_{op_1}} = \langle \underbrace{\{onRoadLine^{op} \leftarrow not\ crossLine^{op}\}}_{S}, \underbrace{onRoadLine^{op}}_{op} \rangle$$

[5] Please, note that in atom:$speed > 0kmh^{co}$ the symbol $>$ does not belong to the underlying language, it is a semantic interpretation of a world observation

Conflicts among operative hypotheses At some point in the deliberation, an agent can build a number of operative hypotheses about its beliefs, these can be conflicting each other invalidating or supporting other. This process has been used in argumentation theory for endowing non-monotonic reasoning to agents.

Definition 4 (Attack relationship between hypotheses).
Let $H_A = \langle R_A, op_A \rangle$, $H_B = \langle R_B, op_B \rangle$ be two operative level hypotheses such that $ASP(R_A) = \langle T_A, F_A \rangle$ and $ASP(R_B) = \langle T_B, F_B \rangle$; with $R_A, R_B \subseteq R$ i.e., hypotheses with related information. We can say that H_A attacks H_B if one of the following conditions holds: 1) $op_A \in T_A$ and $\neg op_A \in T_B$; and 2) $op_A \in T_A$ and $op_A \in F_B$. $Att(\mathcal{H})$ denotes the set of attack relationships among hypotheses belonging to a total set of possible built hypotheses $\mathcal{H}$.

In argumentation theory literature, Dung in [9] introduced patterns of selection for arguments, the so called *argumentation semantics* which are formal methods to identify conflict outcomes for sets of arguments. The sets of arguments suggested by an argumentation semantics are called *extensions* which can be regarded as conflict-free and consistent explanations. In our approach, using an argumentation semantics to a set of hypotheses (at any level), for instance in the operative level: $SEM(Att(\mathcal{H}_{op}), \mathcal{H}_{op})$ the function SEM returns "the best" explanations for the current situation, where $\mathcal{H}_{op}$ denotes the set of all operative hypotheses that can be built from P. We can denote $SEM(AF_{op}) = \{Ext_1, \ldots, Ext_m\}$ as the set of m extensions generated by an argumentation semantics *w.r.t.* an *argumentation framework* formed by operational level hypotheses $AF_{op} = \langle \mathcal{H}_{op}, Att_{op} \rangle$. Sets of justified conclusions from the argumentation process can be defined as follows:

Definition 5. *(Justified conclusions)*
Let P be an extended logic program capturing an activity; let $AF_{op} = \langle \mathcal{H}_{op}, Att_{op} \rangle$ be the resulting argumentation framework from P and SEM be an argumentation semantics. If $SEM(AF_{op}) = \{Ext_1, \ldots, Ext_m\}, (m \geqslant 1)$, then: $\mathsf{Concs}(E_i) = \{\mathsf{Conc}(H) \mid H \in E_i\} (1 \leqslant i \leqslant m)$ and $\mathsf{Output} = \bigcap_{i=1\ldots n} \mathsf{Concs}(E_i)$.

In the remainder of this paper, we use subscripts with this functions to define the deliberation context, *e.g.,* Output_{op} indicates an output set of a deliberation process in the operative level of an activity.

Proposition 1. Concs *from operative hypotheses are candidate beliefs for an agent.*

Proposition 2. Output *in the operational level suggests an unambiguous belief for an agent.*

3.2 Deliberation in the objective level

Objective hypotheses captures the notion of consistent agent desires, describing necessary conditions to achieve a goal as objective. In this sense, an objective hypothesis is composed by operative level hypotheses directed to a goal, more formally:

Definition 6 (Objective hypothesis). *Let $\mathcal{A} = \langle Go, Ac, Op, Co \rangle$ be an agent activity. Let $S \subseteq P$ be a subset of an extended logic program; let $g \in Go$ be a goal and let $R = \mathsf{rel\_rules}(S, g)$ be the set of clauses related to g. Let Output_{op} be the output of the deliberation process in the operative level*[6]*. An objective hypothesis is a tuple $H_{ob} = \langle R^{'}, g \rangle$ if the following conditions hold:*

1. *$ASP(R) = \langle T, F \rangle$ such that $g \in T$.*
2. *R is minimal w.r.t. the set inclusion, satisfying condition 1.*
3. *$R^{'} = R \cup \mathsf{Output}_{op}$.*
4. *$\nexists\, g \in \mathcal{L}_P$ such that $\{g, \neg g\} \subseteq T$ and $ASP(R) = \langle T, F \rangle$.*

where $Op, Co, Go \subseteq R$. Output_{op} is a set of unambiguous beliefs in the operational level. $\mathcal{H}_{ob}$ will denote the set of all the objective hypotheses that can be built from P.

In Definition 6, R is extended with a set of unambiguous belief from the operative level: Output_{op} *i.e.,* a number of facts from the operative level are added to the subset of clauses related to a given goal. This bottom-up building approach has two advantages: 1) restricts the search space for building objective desires; and 2) limits the generation of agent's desires by constraining the output of the deliberation process to sets of unambiguous beliefs using Output_{op}.

Example 4. Let us continue with Example 1. Let us assume the following output from the deliberative process in the operative level: $\mathsf{Output}_{op} = \{onRoadLine^{op}\}$ (see Example 3), an operative hypothesis can be built:

$$H_{ob_1} = \langle \underbrace{\{keepRoute^{g} \leftarrow onRoadLine^{op} \wedge not\ carNear^{op} \wedge \ldots;\ onRoadLine^{op} \leftarrow not\ crossLine^{op}; \underbrace{onRoadLine^{op} \leftarrow \top}_{\mathsf{Output}_{op}}\}}_{R'},\ \underbrace{keepRoute^{g}}_{g} \rangle$$

Where $\leftarrow \top$ is a clause that always evaluates true, so called *fact*.

In the hierarchical structure of AT, goals can be composed by other goals inducing the notion of a sub structure of an objective hypothesis, as follows:

Definition 7. *Let $H_{ob_C} = \langle R_C, ob_C \rangle$, $H_{ob_D} = \langle R_D, ob_D \rangle$ be two objective hypotheses. H_{ob_C} is a sub-objective hypotheses of H_{ob_D} if and only if $R_C \subset R_D$.*

Similarly to operative hypotheses, among objective hypotheses attack relationships may exist. Moreover, *inter-level* attacks, *i.e.,* hypotheses from a level attacking other hypotheses in different level, can also occur due to the bottom-up deliberation process that is performed using AT approach.

Proposition 3. Output *in the objective level suggests unambiguous desires for an agent.*

Proposition 4. *Agent desires can be composed by operative and objective hypotheses, i.e. desires can be formed by other desires or consistent beliefs.*

[6] Assuming that $AF_{op} = \langle \mathcal{H}_{op}, Att_{op} \rangle$ is the resulting argumentation framework obtained from R and $SEM(AF_{op}) = \{Ext_1, \ldots, Ext_m\}, (m \geqslant 1)$ is the set of extensions suggested by an argumentation semantics SEM

3.3 Deliberation in the intentional level

A third type of hypotheses allowing to an agent deliberate about how to reach a goal by executing an action under certain circumstances is proposed.

Definition 8 (Intentional hypothesis). *Let $\mathcal{A} = \langle Go, Ac, Op, Co\rangle$ be an agent activity. Let $S \subseteq P$ be a subset of an extended logic program; let $g \in Go$ and $acc \in Ac$ be a goal and an action; let $R^{'} = \mathsf{rel\_rules}(S, acc)$ be the set of clauses related to acc. Let Output_{obj} be the output of a deliberation process in the objective level[7]. An intentional hypothesis is a tuple $H_{in} = \langle R^{''}, g, acc\rangle$ if the following conditions hold:*

1. *$ASP(R^{''}) = \langle T, F\rangle$ such that $g \in T$.*
2. *$R^{''}$ is minimal w.r.t. the set inclusion satisfying 1.*
3. *$R^{''} = R^{'} \cup \mathsf{Output}_{obj}$.*
4. *$\nexists\ g, acc \in \mathcal{L}_P$ such that $\{g, \neg g\} \subseteq T$, $\{acc, \neg acc\} \subseteq T$ and $ASP(R^{''}) = \langle T, F\rangle$.*

where $Op, Co, Acc, Go \subseteq R^{'}$. Output_{obj} is a set of unambiguous desires in the objective level. $\mathcal{H}_{in}$ will denote the set of all the intentional hypotheses that can be built from P.

Similarly to the deliberation process in operative and objective levels, Definition 8 establishes a bottom-up process using previous deliberations but including information *how* to achieve the given goal.

Example 5. Example 1 continuation. Following the bottom-up approach, desires and beliefs from previous deliberative process are added to the related rules of action $throttleUp^{acc}$. Using Definition 8 an intentional hypothesis can be built:

$$H_{in_1} \langle \underbrace{\{arriveDestination^{g} \leftarrow keepRoute^{g} \ \wedge\ throttleUp^{acc};\ keepRoute^{g} \leftarrow onRoadLine^{op} \wedge not\ carNear^{op} \wedge speed > 60kmh^{co};\ onRoadLine^{op} \leftarrow not\ crossLine^{op}; \underbrace{onRoadLine^{op} \leftarrow \top}_{\mathsf{Output}_{op}}; \underbrace{keepRoute^{g} \leftarrow \top}_{\mathsf{Output}_{obj}}\}}_{R''}, \underbrace{throttleUp^{acc}}_{acc}, \underbrace{arriveDestination^{g}}_{g}\rangle$$

The intentional hypothesis H_{in_1} has an action $throttleUp^{acc}$ that when is executed under certain operations-conditions, hypothetically the agent will achieve the goal $arriveDestination^{g}$.

An argument-based deliberation in the intentional level of an activity, can suggest sets of consistent intentions in an agent. Output_{in} can be defined as a set of conclusive hypotheses supporting means (actions) to reach goals. As a result of this hierarchical structure and similarly in the objective and operative levels, *sub-intentional hypotheses* can be also defined (we omit these formal definition).

Proposition 5. *Output_{in} suggests unambiguous intentions for an agent. Concs_{in} are candidates for agent intentions.*

[7] Similarly Definition 6, assuming that $AF_{obj} = \langle \mathcal{H}_{obj}, Att_{obj}\rangle$ is the resulting argumentation framework obtained from $R^{'}$ and $SEM(AF_{obj}) = \{Ext_1, \ldots, Ext_m\}, (m \geqslant 1)$ is the set of extensions suggested by an argumentation semantics SEM

4 A tool for argument-based deliberation on complex activities

In this section, we briefly describe the tool[8] for lack of space. The first module in Figure 2 evaluates the inference feasibility of an atom considering if an atom belongs to the head of a rule or not. We obviate present this algorithm for a lack of space and simplicity of the process.

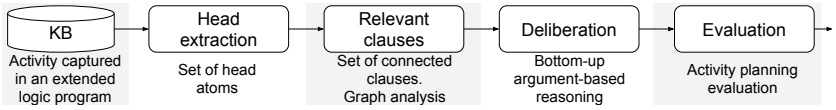

Fig. 2. Deliberation tool modules and implementation notes.

Relevant clauses search is one of the key components in our approach. For a lack of space we cannot present this algorithm. Nevertheless we implement this in our tool using a graph library for detect connected components treating the logic program as a graph. *Deliberation* module takes a mapping between heads and their relevant rules and generates hypotheses first in the operative level, considering only atoms that are in the heads of rules which belong to operational level rules. Then, a semantic argumentation is applied using an external tool, a modification of the WizArg tool [12]. The output set is stored and the algorithm for selecting heads is again applied to obtain new facts which are added to the subprograms. This process is repeated for the objective and intentional layers of the activity. Based on the notion of a *semantic-based construction of arguments* [13] we developed a similar tool using DLV [19]. In Algorithm 1 line 5, MIN() is a function returning the minimal set *w.r.t.* the evaluated answer-set. Let us note that in the same line, ASP() function can be implemented using well-founded or stable semantics. In our implementation, we use the well-founded semantics evaluation provided by DLV (option -WF).

Algorithm 1: Deliberation at operational level

input : $map(atom, \mathsf{rel\_rules}(P, atom))$
output: Output_{op}

1 Let R, Hyp_{op} and out be empty sets
2 Let `ASP()` be an answer-set evaluation of a rule set
3 Let `OUTPUT()` be a function following Definition 5
4 Let `SEM()` be an argumentation semantics evaluation of hypotheses set
5 Let `MIN()` be a function selecting the minimal set
6 **foreach** $atom \in map(atom, \mathsf{rel\_rules}(P, atom))$ **do**
7 $\quad R =$ `MIN(ASP(`$\mathsf{rel\_rules}(P, atom)$`))`
8 $\quad$ **if** $R \neq \{\varnothing\}$ **then**
9 $\quad\quad Hyp_{op} = Hyp_{op} \cup \langle R,\ atom \rangle$
10 $\quad$ **end**
11 **end**
12 $out =$ `OUTPUT(SEM(`Hyp_{op}`))`
13 Returns out

[8] Sources and manual instructions of the tool can be download in: `https://github.com/esteban-g/recursive_deliberation`

5 Discussion

In this paper, the research question: how a software agent can look ahead for the next goal to execute when current and past events are considered? is addressed. For this purpose, we propose a bottom-up process for building consistent hypotheses allowing to an agent deliberate about what action (or set of actions) take to accomplish a goal (or set of goals). Current and past events are considered here no as temporal occurrences, *i.e.* considering the time when actions are performed (*e.g.* temporal reasoning), but as the "classical" notion of *fluents* [21]. Argument-based hypotheses are built to characterize mental states of the agent framed on a particular activity. Knowledge representation structure of the agent is based on an activity theory perspective, which allows us to clearly define the role of goals and actions *w.r.t.* an activity. In different approaches of practical reasoning using a Belief-Desire-Intention model, some agent's goals have analogous interpretation than desires[9]. In our approach, a well-known theory for activity analysis defines an interpretation of actions, goals, operations and conditions. Belief, desires and intentions of the agent are built upon an activity. In this sense, our approach is close to the Kautz *plan recognition* [16, 17], where a hypothetical reasoning method is proposed in which an agent tries to find some set of actions whose execution would entail some goal.

There are key points to highlight why we consider this framework a valuable resource to be considered in practical reasoning: 1) *granularity of actions and goals*, in a number of approaches in computer science, actions are considered atomic processes directed to another atomic structure, the goal (see [14, 28] reviewing agent theories). In different approaches of activity recognition, deviations in what is considered a "normal" activity have been amply investigated (see [26] as survey). In our approach, granularity in acts is the key for our bottom-up agent deliberation. 2) *Activity as a hierarchical dynamic structure*. Essential in our approach is activity dynamics. Roughly speaking, in most of computer science approaches the notion of an activity is statically defined. While this makes it relatively easy to design laboratory experiments, real-world human activities are far more complex and practical agent's activities became compound rather than atomic. Activity theory establishes a valuable approach for explaining real-world activity dynamics, *e.g.,* activities changing in time.

The closest approaches of our bottom-up deliberation are formal models for reasoning about desires, generating desires and plans for achieving them, based on argumentation theory in [2, 24] and [1]. In those approaches, authors propose three frameworks for reasoning about belief, desire and intentions. There are considerable differences between Amgoud *et.al.* and our approach: 1) in [24] an agent has different and independent knowledge bases for beliefs, *desire-generation* rules, and plans; we propose one knowledge base capturing an activity in a logic program. Nevertheless, our approach can deal with multiple concurrent programs given the well known properties of extended-logic programs and ASP semantics (see [6] and [7]); 2) in [1] the argument-based structure of actions and desires can lead to inconsistencies of the form: $\{desire \leftarrow desire\}$[10]; 3) an action in [1] is a tuple $\langle desire, Plan \rangle$ (original nota-

[9] *e.g.* the so called, "potential desires" and "potential initial goals" in [1, 2]

[10] In [1] Definition 4 it is state that "Note that each desire is a sub-desire of itself".

tion is different); in our approach *action* is an established notion in social sciences of a higher level act; 2) in [24] and [1], deliberation process is linked to the semantic meaning of atoms; we proposed our bottom-up approach considering an activity as a reference background framework where beliefs can change not only under more evidence or information (Definition 2) but also by a process called *automatization* in AT literature, where actions transform in operations[11]. A key advantage of our approach is the ability of maintain a *reasoning focus*, *e.g.,* in program P of Example 1 a clause about checking information about social activities: $verifySocialNet^{g} \leftarrow touchScreen^{op} \ \wedge internetAvailab^{co}$ does not affect the inference about driving, the so-called *conflict propagation* [10] or *contamination* [5].

6 Conclusions

We present a formalization about an argument-based deliberation method for building explanations about current and past agent's events. Knowledge of the agent is represented using an activity-theoretical framework captured in an extended logic program. A bottom-up progressive approach for building structured beliefs, desires and intentions is formalized and implemented. We present algorithms used for developing our deliberation tool which we released as open-source. This is a first step in the integration of an activity-theoretical approach for knowledge representation of software agents. In our future work we want to investigate the process of change in complex software agent's activities similarly as is analyzed in social sciences. In this manner, an agent can re-orient plans when actions become operations, *e.g.,* when a software agent learns an activity by imitation or using human support, then such activity changes.

References

1. L. Amgoud. A formal framework for handling conflicting desires. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 552–563. Springer, 2003.
2. L. Amgoud and S. Kaci. On the generation of bipolar goals in argumentation-based negotiation. In *International Workshop on Argumentation in Multi-Agent Systems*, pages 192–207. Springer, 2004.
3. K. Atkinson and T. Bench-Capon. Practical reasoning as presumptive argumentation using action based alternating transition systems. *Artificial Intelligence*, 171(10):855–874, 2007.
4. T. Bench-Capon, P. Dunne, T. Bench-Capon, and P. E. Dunne. Argumentation in artificial intelligence. *Artificial Intelligence*, 171(10):619–641, 2007.
5. M. W. Caminada, W. A. Carnielli, and P. E. Dunne. Semi-stable semantics. *Journal of Logic and Computation*, pages 1–48, 2011.
6. J. Dix. A classification theory of semantics of normal logic programs: I. strong properties. *Fundam. Inform.*, 22(3):227–255, 1995.
7. J. Dix. A classification theory of semantics of normal logic programs: Ii. weak properties. *Fundam. Inform.*, 22(3):257–288, 1995.

[11] In this paper we do not address automatization, this particular topic is being currently explored by the authors.

8. J. Doyle. Rationality and its roles in reasoning. *Computational intelligence*, 8(2):376–409, 1992.
9. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357, 1995.
10. P. M. Dung and P. M. Thang. Closure and consistency in logic-associated argumentation. *Journal of Artificial Intelligence Research*, 49:79–109, 2014.
11. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New generation computing*, 9(3-4):365–385, 1991.
12. I. Gómez-Sebastià and J. C. Nieves. Wizarg: Visual argumentation framework solving wizard. In *Artificial Intelligence Research and Development conf.*, pages 249–258, Amsterdam, Netherlands, 2010. IOS Press.
13. E. Guerrero, J. C. Nieves, and H. Lindgren. Semantic-based construction of arguments: An answer set programming approach. *International Journal of Approximate Reasoning*, 64:54 – 74, 2015.
14. N. R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Autonomous agents and multi-agent systems*, 1(1):7–38, 1998.
15. V. Kaptelinin and B. A. Nardi. *Acting with Technology: Activity Theory and Interaction Design*. Acting with Technology. MIT Press, 2006.
16. H. A. Kautz. Chapter 2 - a formal theory of plan recognition and its implementation. In J. F. Allen, , H. A. Kautz, , R. N. Pelavin, , and J. D. Tenenberg, editors, *Reasoning About Plans*, pages 69 – 125. Morgan Kaufmann, San Francisco (CA), 1991.
17. H. A. Kautz and J. F. Allen. Generalized plan recognition. In *Proceedings of the 5th National Conference on Artificial Intelligence. Philadelphia, PA, August 11-15, 1986. Volume 1: Science.*, pages 32–37, 1986.
18. K. Kuutti. Activity theory as a potential framework for human-computer interaction research. *Context and consciousness: Activity theory and human-computer interaction*, pages 17–44, 1996.
19. N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The dlv system for knowledge representation and reasoning. *ACM Transactions on Computational Logic (TOCL)*, 7(3):499–562, 2006.
20. A. N. Leontyev. Activity and consciousness. *Moscow: Personality*, 1974.
21. J. McCarthy and P. Hayes. *Some philosophical problems from the standpoint of artificial intelligence*. Stanford University USA, 1968.
22. J. E. Naranjo, M. A. Sotelo, C. Gonzalez, R. Garcia, and T. De Pedro. Using fuzzy logic in automated vehicle control. *IEEE Intelligent Systems*, 22(1):36–45, 2007.
23. M. J. O'Donnell. *Introduction: Logic and Logic Programming Languages*, volume 5, *Logic Programming*, chapter 1. Oxford University Press, 1998.
24. I. Rahwan and L. Amgoud. An argumentation based approach for practical reasoning. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 347–354. ACM, 2006.
25. A. S. Rao and M. P. Georgeff. Modeling rational agents within a bdi-architecture. *KR*, 91:473–484, 1991.
26. P. Turaga, R. Chellappa, V. S. Subrahmanian, and O. Udrea. Machine recognition of human activities: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1473–1488, 2008.
27. A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *J. ACM*, 38(3):619–649, July 1991.
28. M. Wooldridge and N. R. Jennings. Agent theories, architectures, and languages: a survey. In *International Workshop on Agent Theories, Architectures, and Languages*, pages 1–39. Springer, 1994.