Jumping to conclusions

Ján Šefránek

Institute of Informatics, Comenius University
842 15 Bratislava, Slovakia
e-mail: sefranek@fmph.uniba.sk

Abstract. This paper is aiming at a re-examination of the idea of jumping to conclusions. Our aspiration is to create a (formal) theory of reasoning with two features: A quick, incomplete and error-prone reasoning is enabled on the one hand. Rigorous reasoning specified by various logical systems is accessible on the other hand. The both forms of reasoning are connected in a unified (formal) framework.
Jumping to conclusions is based on belief patterns, on sets of literals. There is not a necessary correspondence between a belief pattern and a knowledge base. Reasoning is fired by some events. The events are represented by some messages of a communication language.
An abstract formal framework satisfying the above mentioned properties is defined. The framework is "implemented" in the environment of dynamic logic programming.

1 Introduction

The seminal idea of jumping to conclusions [11] provided an important influence on the investigation of common sense and knowledge representation in the last two decades. Many deep results emerged in the field of logical foundations of common-sense reasoning.[1] The state of the art was shifted considerably, [10]. Significant progress in the domain led to some negative results. It has been shown that the inference specified by the formalizations of common-sense reasoning is significantly harder than classical, monotonic inference, see [4, 5] (and many others).

This paper is aiming at a re-examination of the idea of jumping to conclusions. The basic formalizations of non-monotonic reasoning use the idea of "jumping to conclusions" as characterized for example in [7]: "A reasoning agent can be said to be 'jumping to conclusions' whenever it draws conclusions that are not deductively justified by the premises". The formalizations of non-deductive inference satisfy strict criteria of the logical rigor and thoroughness. It is paid by hardness of the problems inherent in these formalizations.

[1] The field may be characterized by the label of non-monotonic reasoning. We will use both terms – common-sense reasoning and non-monotonic reasoning – interchangeably. The term "hypothetical reasoning" is also appropriate (at least for the author).

We change the perspective: People usually do not use much of inference. The consequences are “immediately seen”. Some patterns are used when jumping to conclusions. We distinguish between the patterns and the knowledge explicitly expressed by some sentences of a language. We assume that a reasoning based on the patterns may be error-prone. Justifications, arguments are used only if some problems appear (when “jumping to conclusions”). *A formal theory of a combined use of patterns and of justification (argumentation, inference) in (commonsense) reasoning* is presented in the paper. Our goal is to preserve the idea of a quick jumping to conclusions together with a possibility to combine it with a rigorous, thorough, complete reasoning (both deductive and non-monotonic) based on a logical discipline. The combination is “embodied” by two kinds of reasoners – ideal reasoners (corresponding to logics) and reasoning agents (characterized by jumping to conclusions, using some belief patterns, and sometimes “consulting” the ideal reasoners). A shift from ”jumping” to ”argumentation” is fired by some events.

The technical core of the paper use the paradigm of dynamic logic programming [1]. The paradigm provides a formalization of knowledge evolution (an essential feature from the non-monotonic reasoning point of view). The paper is organized as follows. In the Section 2 we prepare the basic intuitions and some decisions concerning the proposed formalization. Next, in the Section 3, some general formal concepts are introduced. The general notions are “implemented” in the environment of dynamic logic programming in the Section 4. Some open problems are presented in the Section 5.

2 Reasoning agents (the basic decisions)

Our basic aspiration is to create a (formal) theory of reasoning with two features:

– We distinguish between quick and error-prone reasoning on the one hand and a rigorous logical specification of reasoning on the other hand. The later we “embody” as *ideal reasoners*, the former as *reasoning agents*.
– We connect both forms of reasoning in a unified framework. The framework enables a reasoning agent to jump to conclusions. If some problems appear the agent may call (consult) an ideal reasoner.

In this section we collect some motivations for our approach, a verbal description of the basic constructions/notions and some decisions/requirements concerning the formalization proposed in this paper. First, we introduce an example.

Example 1 Let us imagine a little boy. He wants to buy a cake (4 SKK)[2] and a bottle of mineral water (3 SKK). Consider his desire as an event firing a reasoning task. The goal of the task is to compute the needed amount of money. One of boy’s belief patterns is activated, 4 + 3 is 6 according to the pattern. The boy gives 6 SKK to a woman selling in a buffet. He receives a message of

[2] Slovak crowns

disagreement immediately. Another reasoning task is fired by the message. May be, the belief pattern is modified consequentially. □

Description 1 (Belief Pattern) Quick reasoning, 'jumping to conclusions' is based on *belief patterns*, some clusters of elementary beliefs. Each reasoning agent uses a set of belief patterns. The patterns are activated by some events.

We represent belief patterns by sets of literals. If an agent evaluates a formula w.r.t. a belief pattern, it "sees immediately" which literals are true (false, unknown, contradictory) w.r.t. the belief pattern. □

Description 2 (Event) A reasoning agent is embedded into an environment. Its reasoning is activated by some events, signals from the environment or by some inner events (intentions, desires).

The events are represented in this paper by messages of a communication language. The messages sent by an agent to itself are not excluded. □

We now describe what we understand by a reactive agent in this paper. Further we describe our intuitions concerning reasoning agents.

Description 3 (Agent) A (reactive) agent behaves as a function: it receives a message (a query) and responds by a message. The query is evaluated with respect to the current belief pattern and the computed value is returned in the response.

An agent is behaving (and reasoning) in an environment. We identify the environment with a set of agents exchanging some messages. □

Reasoning agents are agents, endowed with a special structure.

Description 4 (Reasoning Agent) A reasoning agent has an access to a set of belief patterns.

Its reasoning is based on a current belief pattern. The pattern is selected from the set of belief patterns. The selection depends on a way how belief patterns are connected to some events. When an event occurs, a (less or more) appropriate belief pattern is activated. An agent is able to modify the current belief pattern.

An agent is able to respond to some events immediately using its belief patterns and to engage in a more detailed argumentation when needed. The immediate (and error-prone) responding is called jumping to conclusions. Some of the events signalizes that there appear some problems when jumping to conclusions. These events initialize a call to an ideal reasoner.

We assume that each reasoning agent has an access to a knowledge base and it is able to extend the knowledge base (it is able to acquire new information). □

Some additional intuitions concerning events resulting in a call to ideal reasoners are presented in the Section 4.

3 A general frame

We assume a *background language* $\mathcal{L}$.

Definition 2 (Belief Pattern) Let L be the set of all literals of the language $\mathcal{L}$.

A *partial interpretation* of the language $\mathcal{L}$ is a subset of L. A *belief pattern* (w.r.t. $\mathcal{L}$) is a partial interpretation (of $\mathcal{L}$). □

Four-valued interpretations are assumed. A partial interpretation may contain both an atom A and the literal $\neg A$. On the other side, a partial interpretation $\mathcal{I}$ such that neither $A \in \mathcal{I}$ nor $\neg A \in \mathcal{I}$ is admissible.

In this section we assume that satisfiability relation $\models$ is (somehow) defined. In the next section we will define the satisfiability for the language of generalized logic programs.

Definition 3 (Communication Language) The *communication language* $\mathcal{L}_C$ over a language $\mathcal{L}$ contains the *elementary expressions* as follows:

- $ASK(q)$, where q is a conjunction of literals,[3]
- $RESPOND(a, v)$, where a is an elementary expression of the form $ASK(q)$ and $v \in \{t, f, u, c\}$ is a truth value,[4]
- $MORE(a)$, where a is as above,
- $TELL(S)$, where S is a set of formulae,
- $ATTACK(\phi)$, where ϕ is a formula,
- $JUSTIFY(l, \phi)$, where l is a literal and ϕ is a formula,
- $VIEW(S)$, where S is as above.

□

Of course, also other elementary expressions may be defined. The use of the elementary expressions is described later, in the Section 4 (in the context of dynamic logic programming).

Definition 4 (Agent) An *agent* is a function from the set of all expressions of the form $ASK(q)$ to the set of all expressions of the form $RESPOND(a, v)$. □

Definition 5 (Reasoning Agent) Let $\mathcal{E}$ be the set of all elementary expressions from $\mathcal{L}_C$, $\mathcal{Q}$ be the set of all elementary expressions of the form $ASK(q)$ (queries), $\mathcal{R}$ be the set of all elementary expressions of the form $RESPOND(a, v)$ (responses).

A *reasoning agent* β is a tuple of the form $(\mathcal{P}, F_{eval}, \rho_b, Arg)$, where

- $\mathcal{P}$ is a set of belief patterns,

[3] A dependency on [9] is declared.

[4] Only yes/no queries are assumed, for the simplicity; admissible responses are true, false, unknown and contradictory.

- $F_{eval} : \mathcal{Q} \times \mathcal{P} \longrightarrow \mathcal{R}$ is a function that evaluates queries,
- $\rho_b \subseteq \mathcal{P} \times \mathcal{E} \times \mathcal{P}$ is a relation such that for each triple $(u, e, w) \in \rho_b$ holds that w is a candidate for the current belief pattern if the previous belief pattern u and the elementary expression e are given,[5]
- *Arg* is an *argumentation unit*, an encapsulated set of functions.[6]

$\Box$

Remark 6 For some classes of expressions there is exactly one w such that $(u, e, w) \in \rho_b$: If e is of the form $RESPOND(a, v)$ or $MORE(a)$, then $w = u$. If e is of the form $ASK(q)$ or $VIEW(S)$ we will use the functional notation: $\rho_b(u, e) = w$. $\Box$

Definition 7 (Message) A *message* is a triple (α, β, σ), where α, β are agents (the sender and the receiver, respectively) and σ is a sequence of elementary expressions of $\mathcal{L}_C$. $\Box$

Definition 8 (Protocol) A *protocol* is a pair of the form (Σ, F), where Σ is a sequence of messages in $\mathcal{L}_C$ and F is a function. $\Box$

The function implements an appropriate reasoning task when Σ happens. Intuitively, after an exchange of a series of messages (after some events) the receiver of the last message of Σ should compute F in order to accomplish an appropriate reasoning task. Some of the reasoning tasks will be presented in the Section 4. Some of them represent "ideal reasoners".

Definition 9 (Ideal Reasoner) Let $\mathcal{S}$ be the set of all formulae of $\mathcal{L}$ and $2^{\mathcal{S}}$ be the set of all subsets of $\mathcal{S}$.

An *ideal reasoner* is an operator $Cn : 2^{\mathcal{S}} \longrightarrow 2^{\mathcal{S}}$. $\Box$

A very simple idea of an ideal reasoner is used here: we assume only that it is able to manipulate sets of sentences.

4 Dynamic logic programming framework

4.1 Preliminaries

We consider *generalized logic programs* [1] in this section. A propositional language $\mathcal{L}$ is assumed. A *generalized clause* (abbreviated to *clause* or *rule* hereafter) is an expression of the form $L \leftarrow L_1, \ldots, L_k$, $k \geq 0$, where L, L_i are literals (propositional letters or their negations). Only default negation is used in this section. We will denote L also by $head(\phi)$ and the conjunction $L_1, \ldots, L_k$ by $body(\phi)$. A generalized logic program (abbreviated to *program* hereafter) is a

[5] The current belief pattern is selected in a non-deterministic way by the reasoning agent. We assume that the sequence of current belief patterns is initialized somehow.
[6] See the Section 4.

(non-empty) set of clauses. The fact that programs allow default negation in heads of the rules enables to specify both insertions and deletions by the rules of a program.

The general concepts from Section 3 are specialized in a straightforward way. We now define how clauses are evaluated in a belief pattern.

Definition 10 Let L be a literal, $\overline{L}$ be its complement and w be a belief pattern. Then

- $w \models L$ iff $L \in w$ and $\overline{L} \notin w$,
- $w \not\models L$ iff $\overline{L} \in w$ and $L \notin w$,
- $w \models^c L$ iff $\overline{L} \in w$ and $L \in w$,
- $w \models^u L$, otherwise.

Let q be a query of the form $\leftarrow L_1, \ldots, L_k$. Then

- $w \models q$ iff $w \models L_i$, for each i,
- $w \not\models q$ iff $w \not\models L_i$, for some i,
- $w \models^c q$ iff $w \models^c L_i$, for some i,
- $w \models^u q$, otherwise.

Let ϕ be a clause of the form $L \leftarrow L_1, \ldots, L_k$. Then

- $w \models \phi$ iff $w \models L_1, \ldots, L_k$ and $w \models L$,
- $w \not\models \phi$ iff $w \models L_1, \ldots, L_k$ and $w \not\models L$,
- $w \models^c \phi$ iff $w \models L_1, \ldots, L_k$ and $w \models^c L$,
- $w \models^u \phi$, otherwise.

Let S be a program. Then

- $w \models S$ iff for each $\phi \in S$, $w \models \phi$,
- $w \models^c S$ iff there is $\phi \in S$ such that $w \models^c \phi$,
- $w \not\models S$ iff there is $\phi \in S$ such that $w \not\models \phi$ and for no $\phi \in S$ holds $w \models^c \phi$,
- $w \models^u S$, otherwise.

We will use also $\models^t$ instead of $\models$ and $\models^f$ instead of $\not\models$. $\Box$

The basic construction used in dynamic logic programming (DyLoP hereafter) is as follows [1]. Two programs are given: S (the original program) and U (the updating program). A new program $S \oplus U$ (the updated program) is specified (computed) in DyLoP. This construction is generalized to sequences of program updates $S \oplus U_1 \oplus \cdots \oplus U_n$, we will use also the notation $\oplus\langle S, U_1, \ldots, U_n\rangle$. More complicated structures on the sets of programs may be defined, for example graphs, [6]. However, in this section we consider only sequences and the basic case $S \oplus U$ is considered usually.

The paradigm of DyLoP provides an appropriate tool for a representation of dynamically changing and modular knowledge. We assume in this section that a reasoning agent has an access to a knowledge base KB, to a set of programs. Each of the programs may represent a viewpoint, a new cluster of knowledge etc.

We assume that each program in KB has at least one stable model. An insertion of a new program S into KB is allowed. For each sequence Σ of programs from KB a modified program $\oplus\Sigma$ is defined.

We now come to the technical core of the paper, to an "implementation" of reasoning-agent-functions in the DyLoP environment. As a side-effect, a specification of the "behavior" of elementary expressions from $\mathcal{L}_C$ is provided.

A class of protocols is defined in the following subsections of this section. The protocols specify a behavior of a reasoning agent. They may be selected and used by a reasoning agent in a non-deterministic way.

Remark 11 We assume in the following that each current belief pattern is computed using ρ_b. Suppose that the previous belief pattern u and e, the last elementary expression of the sequence of messages in the relevant protocol Σ, be fixed. Then the computation selects one belief pattern w from the set of triples $\{(u, e, w) : (u, e, w) \in \rho_b\}$.

Our model of reasoning is a non-deterministic one: a context (represented by a set of messages) may to fire various reasoning tasks (and to select various current belief patterns). Often the current belief pattern remains unchanged. Sometimes the change is straightforward. A specification of the current belief pattern computation is given explicitly in the following only if there is a special reason for giving it. $\Box$

4.2 Quick reasoning

Definition 12 (Responding) Let $\Sigma = \langle(\alpha, \beta, \langle ASK(q)\rangle)\rangle$ be a sequence of messages and w_0 be the current belief pattern. We define a protocol (Σ, F_{eval}):

$$F_{eval}(ASK(q), w) = RESPOND(ASK(q), x) \text{ iff } w \models^x q,$$

where $w = \rho_b(w_0, ASK(q))$ and $x \in \{t, f, c, u\}$. $\Box$

The protocol specifies a behavior of a reasoning agent β receiving a query.

Example 13 Let us return to the situation of the Example 1.

Suppose that the desire of the little boy is represented by the question "How much crowns I need if I need 4 crowns and 3 crowns?" Only yes/no questions are assumed in our formalization, therefore the question may be $ASK(3 + 4 = 6)$.

Let the activated belief pattern be $\rho_b(w_0, ASK(3 + 4 = 6)) = w = \{1 + 1 = 2, 1 + 2 = 3, \ldots, 1 + 9 = 10, 2 + 3 = 5, \ldots, 2 + 8 = 10, 3 + 4 = 6\}$.

A reasoning task is fired by the question and a response based on the belief pattern is sent (by the boy to himself). If $\Sigma = \langle boy, boy, \langle ASK(3 + 4 = 6)\rangle\rangle$, then

$$F_{eval}(ASK(3 + 4 = 6), w) = RESPOND(ASK(3 + 4 = 6), t).$$

$\Box$

Fact 14 *F_{eval} may be computed in logarithmic time w.r.t. to the size of a belief pattern (if the belief pattern is sorted).* $\Box$

4.3 One-step justification

Consider our running example, but slightly modified.

Example 15 First, suppose that the little boy is right and the woman wants to deceive him. The reasoning of the boy and the communication between the boy and the woman is represented by Σ:

$$
\begin{aligned}
&\Sigma = \\
&\langle(\alpha, \alpha, \langle ASK(3+4=7), RESPOND(ASK(3+4=7), t)\rangle) \\
&(\alpha, \beta, \langle TELL(\{price(\{cake, water\}, 7)\}\rangle), \\
&(\beta, \alpha, \langle ATTACK(price(\{cake, water\}, 7), TELL(\{3+4=8\})\rangle)\rangle.
\end{aligned}
$$

The event represented by an $ATTACK$-expression (accompanied with a $TELL$-expression) fires a reasoning task (a simple justification).

Imagine that the boy decides for a counter-attack, for example as follows: $2+4=6$, therefore $3+4=7$.

In other words, the boy accesses a relevant module of the knowledge base, selects a relevant rule, for example

$$s(Y) + X = s(Z) \leftarrow Y + X = Z$$

or its instance. If the body of the (instantiated) rule is satisfied in the current belief pattern, then the attacked (more precisely, the TELL-ed) sentence, the head of the (instantiated) rule could be considered supported (defended). □

Definition 16 Let w be the current belief pattern and

$$\Sigma = \langle(\beta, \alpha, \langle ASK(l)\rangle), (\alpha, \beta, \langle RESPOND(a, w) = t\rangle), (\beta, \alpha, \langle ATTACK(l)\rangle)\rangle$$

be a sequence of messages, where $a = ASK(l), r = RESPOND(a, w) = t$. We define a protocol (Σ, F_{one}):

F_{one} produce a message for a pair ⟨ literal, belief pattern ⟩:

$$F_{one}(l, w) = JUSTIFY(l, c),$$

if there is a program $S \in KB$ containing a rule c such that $head(c)\theta = l$ for a substitution θ and $w \models^t body(c)\theta$, otherwise F_{one} is undefined. □

We are interested in a quick one-step justification. Therefore we need a quick access to the relevant S (for given w). The selection of the relevant "module" of the knowledge base we implement by a function $mine(w)$.

Definition 17 (Mining) Let w be a belief pattern. We define a function

$$mine : \mathcal{P} \longrightarrow KB,$$

where $mine(w) = S$ and $S \in KB$. □

Fact 18 *Suppose that $Ground(mine(w))$, the ground instance of $mine(w)$, is finite. If KB is indexed by mine, then F_{one} can be computed in linear time with respect to the size of $Ground(mine(w))$.* □

Remark 19 A set of programs may be (and usually is) relevant with respect to a belief pattern w. The definition of *mine* could be modified: $mine'(w) \subseteq KB$ should be satisfied. □

In the case when $F_{one}(l, w)$ is undefined a more detailed argumentation is required, see Subsection 4.6.

4.4 Simple revisions

Let us continue with our running example:

Example 20 Suppose again that the little boy has some problems with the elementary counting.

$$\begin{array}{l} \Sigma = \langle \\ \quad (\alpha, \alpha, \langle ASK(3+4=6), RESPOND(ASK(3+4=6), t)\rangle) \\ \quad (\alpha, \beta, \langle TELL(\{price(\{cake, water\}, 6)\})\rangle), \\ \quad (\beta, \alpha, \langle ATTACK(price(\{cake, water\}, 6)), TELL(\{3+4=7\})\rangle)\rangle. \end{array}$$

The event represented by an $ATTACK$-expression (accompanied with a $TELL$-expression) may fire a modification of the current belief pattern: an acceptance of $3+4=7$ and a rejection of $3+4=6$ (into/from the current belief set).

We repeat that the protocols specifying a behavior of a reasoning agent may be selected and used in a non-deterministic way. (Compare this example with the Example 15.) A selection leading to incorrect reasoning is possible. □

Definition 21 Let w be the current belief set of an agent β,

$$\Sigma = \langle (\beta, \alpha, \langle TELL(\{l\})\rangle), (\alpha, \beta, \langle ATTACK(l), TELL(\{l'\})\rangle)\rangle$$

be a sequence of messages. We define a protocol (Σ, F_r).

$F_r(w) = w'$, where $w' = (w \setminus \{l\}) \cup \{l'\}$.

$(w, TELL(\{l'\}), F_r(w)) \in \rho_b$ and $F_r(w)$ is selected as the new current belief pattern. □

4.5 Unknown or contradictory

Two elementary forms of argumentation (the one-step justification and the simple revision) were discussed in the preceding Subsection. We now present some additional intuitions concerning the events initializing a call to an ideal reasoner, to an argumentation. The problems with reasoning appear, when

- something is *unknown*,

– the beliefs are disordered in a sense (there is an *inconsistency* observed),
– there are repeated *attacks* (from the environment) against some beliefs,
– new information is acquired.

Of course, there are another events firing an argumentation, but we limit ourselves in the sequel to those mentioned above.

We now consider the first two cases: the problems are fired in our formalization by a message of the form $RESPOND(a, x)$, where $x = u$ or $x = c$. A possible stream of argumentation is as follows:

– finding a relevant program (collecting some relevant sentences),
– selecting one stable model of the program,
– modifying the current belief pattern by the stable model,
– finally, responding the given query again (w.r.t. the modified current belief pattern).

We call this *a naive brave reasoning*: one of the alternative meanings is sufficient.

The relevant program is selected by the function *mine*. In order to have a comfortable specification of the naive brave reasoning, we are defining a tool for the selection of one stable model. Remind the assumption that in the knowledge base are collected only the programs with at least one stable model.

Definition 22 (*cred*) The function *cred* selects for given program $S \in KB$ exactly one stable model m , i.e. $cred(S) = m$. □

Fact 23 *Suppose that the computation of cred is preprocessed for each program S from KB. Then $cred(S)$ can be computed in constant time.* □

Remark 24 The idea of preprocessing (compilation) is due to [4]. A Kripke structure associated with a program [12] provides a possible target of a compilation of naive brave stable model reasoning.[7] □

Our next order of business is to define (a kind of) the modification of belief patterns by stable models.

Definition 25 ($\uplus$) Let m be a set of literals and w be a belief pattern. Let l be a literal and $\bar{l}$ be its complementary literal.

Then $w \uplus m = m \cup (w \setminus \{l \in w : \bar{l} \in m\})$. □

We are ready to define repeated responding in order to try to overcome undefinedness or inconsistency.

Definition 26 (Repeated Responding) Let

$$\Sigma = \langle(\alpha, \beta, \langle a\rangle), (\beta, \alpha, \langle RESPOND(a, x)\rangle), (\alpha, \beta, \langle MORE(a)\rangle)\rangle$$

[7] The Kripke structure associated with a program is defined also in the Appendix of the full text of this paper, `http://www.ii.fmph.uniba.sk/ sefranek/online/jump.ps`.

be a sequence of messages, where a is an elementary expression of the form $ASK(q)$, and x is either u or c.

We define a protocol (Σ, F_{rep}), where F_{rep} is a partial function from interpretations to responses.

Let w be the previous belief pattern, $mine(w) = S$, $cred(S) = m$. F_{rep} is applied to the selected current belief pattern $u = w \uplus m$, where $(w, MORE(a), u) \in \rho_b$.

The value of $F_{rep}(u)$ is $RESPOND(a, x)$, where $u \models^x q$. □

We now characterize the type of reasoning specified by F_{rep}.

Definition 27 (Naive Brave Reasoning) Let S be a program, l be a literal, m be a selected stable model of S.

Then $l \in Cn_n(S)$ iff $m \models^t l$. We say that l is a naive brave consequence of S. □

Of course, F_{rep} uses Cn_n: $F_{rep}(u) = RESPOND(ASK(q), t)$ iff $u \models^t l$ iff $l \in Cn_n(S \cup \{l \in w : \bar{l} \notin m\})$.

Fact 28 *If $F_{rep}(u) = RESPOND(a, x)$, then $x \neq c$.* □

Unfortunately, it is not excluded that x in $RESPOND(a, x)$ is again the *unknown* truth value. The technique of the Definition 26 may be used repeatedly. The corresponding protocol is of the form

$$\langle(\alpha, \beta, \langle a\rangle), [(\beta, \alpha, \langle RESPOND(a, x)\rangle), (\alpha, \beta, \langle MORE(a)\rangle)]^+\rangle,$$

where the segment in the brackets $[\ \ ]^+$ may be repeated several times (until a response t or f is reached or the reasoning agent is not interested more in a response). For each new current belief pattern u is computed $n = cred(mine(u))$ and the query is responded w.r.t. $u \uplus n$.

On the other hand, we may use a new function, F_{view}. The function "changes the viewpoint" of the reasoning agent, i.e. it selects a program $U \neq S$. Then we compute the modified program $S \oplus U$ and its stable model is selected as the current belief pattern.

Definition 29 Let S be a program. Let U be another program.

We say that $S \oplus U$ is a naive brave revision of S based on U, if there is $m = cred(U)$, a stable model of U such that $S \oplus U = U \cup \{L \leftarrow L_1, \ldots, L_k \in S : 0 \leq k \ \& \ (m \models L_1, \ldots, L_k \Rightarrow m \not\models \overline{L})\}$. Notation: $S \oplus U = Cn^U_{cr}(S)$. □

Definition 30 Let w be the current belief pattern and Σ be a sequence of the form

$$\langle(\alpha, \beta, \langle a\rangle), [(\beta, \alpha, \langle RESPOND(a, x)\rangle), (\alpha, \beta, \langle MORE(a)\rangle)]^+\rangle.$$

We define a protocol (Σ, F_{view}), where $F_{view}(w) = VIEW(U), U \in KB$, $mine(w) = S$ and $S \neq U$. □

Definition 31 Let w be the current belief pattern and Σ be a sequence of the form

$$\langle(\alpha,\beta,\langle a\rangle),[(\beta,\alpha,\langle RESPOND(a,x)\rangle),(\alpha,\beta,\langle MORE(a)\rangle)]^{+}(\beta,\beta,\langle VIEW(U)\rangle\rangle.$$

Then $\rho_b(w, VIEW(U)) = cred(mine(w) \oplus U)$ and the protocol (Σ, F_{rep}) may be used, i.e. $F_{rep}(cred(mine(w) \oplus U)$ is computed. □

4.6 More sophisticated argumentation

The naive brave reasoning is a fragile and error-prone kind of reasoning. Errors are signalized in this Subsection by a repeated $ATTACK$-message. It may be considered as a trigger for a detailed logical argumentation (based on a rich knowledge base). Similarly, if F_{one} is undefined, the argumentation is triggered. Of course, we are in the domain of computationally hard problems in this moment.

Definition 32 Let w be the current belief pattern, $S = mine(w)$ and

$$\begin{array}{l}\Sigma = \langle\\ \ (\beta,\alpha,\langle ASK(l)\rangle),\\ \ (\alpha,\beta,\langle RESPOND(a,w)=t\rangle),\\ \ (\beta,\alpha,\langle ATTACK(l)\rangle),\\ \ (\alpha,\beta,\langle JUSTIFY(l,c)\rangle),\\ \ (\beta,\alpha,\langle ATTACK(l)\rangle)\rangle\end{array}$$

be a sequence of messages, where $a = ASK(l)$.

We define two protocols:

1. (Σ, F_{br}):
   F_{br} produce a truth-value for a literal: $F_{br}(l) = t$ iff for some stable model m of S holds $m \models^t l$. Otherwise, $F_{br}(l) = f$.
2. (Σ, F_{skept}):
   F_{skept} also produce a truth-value for a literal:
   - $F_{skept}(l) = t$ iff for each stable model m of S holds $m \models^t l$,
   - $F_{skept}(l) = f$ iff for some stable model m of S holds $m \models^f l$,
   - otherwise, $F_{skept}(l) = u$.

□

Fact 33 *For finite propositional programs: a computation of $F_{br}(l)$ is NP-complete and a computation of $F_{skept}(l)$ is coNP-complete.* □

An acquisition of new information (represented by a $TELL(S)$-message) may lead also to a more sophisticated reasoning (as compared with the Examples 15 and 20 and with the Definitions 35 and 21). This kind of reasoning may be implemented in the frame of DyLoP in a natural way. We postpone the discussion of this case to the full text because of the limited size of the paper.

Remark 34 A list of functions from Arg considered in this paper: $\{F_{one}, mine, F_r, F_{rep}, cred, F_{view}, F_{br}, F_{skept}\}$. A list of consequence operators: $\{Cn_n, Cn^U_{cr}\}$.

Of course, the sets are open to modifications in the ongoing research. □

5 Conclusions

A formalization of common-sense reasoning is a rather subtle problem. Let us remind a question from [10]: "Do we already know all the modes of non-monotonic reasoning?". Our goal was to investigate a re-examination of the idea of jumping to conclusions. The kernel of the idea is a combination of a quick reasoning and a strict logical inference.

A considerable part of the current research in non-monotonic reasoning is devoted to the computational aspects of common-sense reasoning (in a broad range of research tasks – from the computational complexity to the new programming paradigms, computational experiments, and also to industrial applications).

Our approach is aiming at a computationally tractable reasoning (reachable in some situations). Simultaneously, the necessity to solve computationally hard problems is preserved (when quick reasoning fails and rigorous solutions, correct from the logical point of view are required). The presented approach is only in the earliest stage of the development. We hope that the framework is a promising one and that the basic decisions presented in the paper are fruitful and able to be developed gradually.

Among the topics of forthcoming papers are:

1. A demonstration that Kripkean semantics for dynamic logic programming may be an appropriate tool for connecting jumping to conclusions and logic-based reasoning.
2. An interesting topic is how to improve the behavior of the functions implementing some reasoning tasks.
   The functions guess some values (belief patterns, programs, etc.). The goal is to made the guess appropriate, adequate. As a consequence a reasoning agent should be more "competent". The functions may be "trained" gradually (people gather the competence stepwise, too), for example using evolutionary algorithms.
3. Action languages could be considered. An insertion of our reasoning agents into dynamically changing domains in the style of [3] is a challenge.

References

1. Alferes, J.J., Leite, J.A., Pereira, L.M., Przymusinska, H., Przymusinski, T.C. *Dynamic Logic Programming.* Proc. KR'98, 1998
2. Alferes, J.J., Leite, J.A., Pereira, L.M., Quaresma, P. *Planning as Abductive Updating.*
3. Baral, C., Gelfond, M. *Reasoning Agents in Dynamic Domains.*
4. Cadoli, M., Donini, F.M., Schaerf, M. *Is intractability of non-monotonic reasoning a real drawback?* Artificial Intelligence 88, 1-2, 215-251

5. Cholewinski, P., Marek, W., Mikitiuk, A., Truszczynski, M. *Experimenting with Nonmonotonic Reasoning.* Proc. 12th ICLP, MIT Press, 1995
6. Dell'Acqua, P., Leite, J., Perez,L. *Evolving multi-agent viewpoints – an architecture.* http://centria.di.fct.pt/ lmp/publications/Biblio.html#sub
7. Flach, P. *An analysis of various forms of 'jumping to conclusions'.* In Jantke, K. (ed.), Proc. 3rd Int. Work. on Analogical and Inductive Reasoning, LNAI, vol. 642, 1992
8. Gelfond, M., Lifschitz, V. *The Stable Model Semantics for Logic Programming.* Proc. 5th ICLP, MIT Press, 1988, 1070-1080
9. Levesque, H. *Foundations of a functional approach to knowledge representation.* Artificial Intelligence 23 (1984), 155-212
10. Marek, W. *NMR – advances, future directions and few questions.* Workshop NMR'2000
11. McCarthy, J. *Epistemological problems of artificial intelligence.* Proc. IJCAI'77, 1038-1044
12. Šefránek, J. *A Kripkean semantics of dynamic logic programming.* In M.Parigot, A. Voronkov (Eds.) Logic for Programming and Automated Reasoning. Springer, 2000, 469-486

A Kripkean semantics for DLP

Definition 35 Let S be a program. A Kripke structure $\mathcal{K}_S$ associated with S is a pair (W, ρ), where:

- $W = Int_S \cup \{w_\perp\}$, W is called the set of possible worlds, Int_S is the set of all partial interpretations of S, $w_\perp$ is the representative of the set of all inconsistent sets of literals,
- ρ is a binary relation on $W \times W$, it is called the accessibility relation and it is composed of two relations: $\rho = \rho_1 \cup \rho_2$, where
  1. the accessibility relation ρ_1 contains the set of all pairs (w, w') such that $w' = w \cup \{head(c_i) : i = 1, \ldots, k\}$, where $c_1, \ldots, c_k$ are (not necessary all) clauses from P such that $w \models body(c_i)$,
  2. if w is not a total interpretation and for no $u \neq w$ there is an edge $(w, u) \in \rho_1$, then $(w, w') \in \rho_2$, where $w' = w \cup \{not\ A : A \notin w\}$.

□

Kripke structures associated with programs enable to identify (and to compute) stable models. Of course, $\mathcal{K}_S$ may be viewed as a graph.

Definition 36 Let $\mathcal{K}_S$ be a Kripke structure associated with a program S. A ρ-path is a sequence σ of edges $(w_0, w_1), (w_1, w_2), \ldots, (w_{n-1}, w_n)$ in $\mathcal{K}_S$ such that each $(w_i, w_{i+1}) \in \rho$.

We say that this σ is *rooted* in w_0 (also w_0-rooted). If there is no ρ-edge (w_n, w) in $\mathcal{K}_S$ such that $w \neq w_n$, we say that σ is *terminated* in w_n (also: w_n is a terminal node of $\mathcal{K}_S$). □

Sometimes we denote paths by the shorthand $\langle w_0, w_1, w_2, \ldots, w_{n-1}, w_n \rangle$. Similarly, a ρ_1-path could be defined.

Now we state conditions for stable models in terms of nodes and paths in $\mathcal{K}_S$.

Definition 37 Let S be a program, σ be an acyclic ρ-path $\langle w_0, w_1, \ldots, w_n \rangle$ from $\mathcal{K}_S$ and $\mathcal{D}$ be the set $\{$*not* $A : A$ is an atom of $S\}$.

We say that σ is *correctly rooted*, if

- either $w_0 = \emptyset$
- or $\emptyset \neq w_0 \subseteq \mathcal{D}$.

□

The following theorems are proven in [12]:

Theorem 38 *Let S be a program, $\mathcal{K}_S$ be the Kripke structure associated with S, $\sigma = (w_0, w_1), (w_1, w_2), \ldots, (w_{n-1}, w_n)$ be an acyclic ρ-path in $\mathcal{K}_S$ terminated in a total interpretation w_n.*

If σ is correctly rooted, then w_n is a stable model of S. □

Theorem 39 *Let m be a stable model of a program S and $\mathcal{K}_S$ be a Kripke structure associated with S.*

There is a correctly rooted and acyclic ρ-path $\sigma = \langle w_0, \ldots, w_n, m \rangle$ in $\mathcal{K}_S$ terminated in m. □

We now come to the belief revision on Kripke structures.

We assume two programs – S, an original program and U, an updating program. Let us recall that the semantics of S and U is specified by $\mathcal{K}_S$ and $\mathcal{K}_U$, respectively. Our goal is to construct a Kripke structure $\mathcal{K}_{S \oplus U}$ over both before mentioned structures.

Definition 40 The path $\sigma = \langle w_0, w_1, \ldots, w_n \rangle$ from $\mathcal{K}_S$ may be connected to a node w from $\mathcal{K}_U$ iff $w_0 \subseteq w$ and $w \cup w_1$ is consistent. □

Definition 41 Let $\sigma = \langle u_0, \ldots, u_n \rangle$ be a ρ-path and w be a node.

Then *connect* σ to w is a partial operation as follows: if σ may be connected to w, then $(w, u_1 \cup w), \ldots, (u_{n-1} \cup w, u_n \cup w)$ is a ρ-path. If for some $i > 1$ holds that $w \cup u_i$ is inconsistent, it is replaced by $w_\perp$ and the rest of the path is removed. □

Definition 42 Let $\mathcal{K}_S$ and $\mathcal{K}_U$ be the Kripke structures associated with non-empty programs P and U, respectively.

We construct $\mathcal{K}_{P \oplus U}$ as follows:

1. each ρ_1-edge from $\mathcal{K}_U$ is an ρ_1-edge of $\mathcal{K}_{P \oplus U}$,

2. for each node w from $\mathcal{K}_U$ and each ρ_1-path $\sigma = \langle u_0, u_1, \ldots, u_n \rangle$ from $\mathcal{K}_S$: connect σ to w,[8]
3. introduce new ρ_2-edges whenever it is possible.

□

[8] The original construction is modified according to [**?**].