Linear Logic, Partial Deduction and Cooperative Problem Solving

Peep Kängas\(^1\), Mihhail Matskin\(^2\)

\(^1\) Norwegian University of Science and Technology
Department of Computer and Information Science
peep@idi.ntnu.no

\(^2\) Royal Institute of Technology
Department of Microelectronics and Information Technology
misha@imit.kth.se

Abstract. In this paper we present a model of cooperative problem solving (CPS). Linear Logic (LL) is used for encoding agents’ initial states, goals and capabilities. LL theorem proving is applied by each agent to determine whether the particular agent is capable of solving the problem alone. If no individual solution can be constructed, then the agent may start negotiation with other agents in order to find a cooperative solution. Partial deduction in LL is used to derive a possible deal. Finally proofs are generated and plans are extracted from the proofs. The extracted plans determine agents’ responsibilities in cooperative solutions.

1 Introduction

It is quite usual in multi-agent systems that an agent needs a help of another agent for performing its tasks and this situation has been considered in several works on cooperative problems solving (CPS). Several attempts were made in order to formalize CPS (see Section 5). Most of them are based on classical or modal logics. In particular, Wooldridge and Jennings [21] provide a formalisation of CPS process where a multi-modal logic is used as a formal specification language. However, since the multi-modal logic lacks a strategy for generating constructive proofs of satisfiability, the formalisation does not lead to direct execution of specifications. Moreover, since modal logics (like classical logic) lack the mechanism for keeping track of resources, it is not possible for agents neither to count nor dynamically update the number of instances of the same object belonging to their internal states.

In order to overcome the mentioned shortages of classical and modal logics we use a fragment of Linear Logic [6] (LL) for CPS. LL provides a mechanism for keeping track of resources (in LL one instance of a formula is distinguished from two or more instances of the formulae) and this makes possible more natural representation of dynamical processes and agents’ internal states.

The cooperative problem solving has been considered to consist of four steps [21]—recognition of potential for cooperation, team formation, plan formation and plan execution.
An important feature of our methodology is that we do not separate team and plan formation into different processes and that negotiation is already embedded into the reasoning. Although this approach does not preserve the accepted structure of CPS, we think that it may be more natural for representing computational aspects of CPS, where team and plan formation processes interact with each other.

Basically, we are applying LL theorem proving for generating a constructive proof representing the first 3 steps of CPS: recognition, team and plan formation. Negotiation is reformulated as distributed proof search. Then a solution, summarising the first 3 steps of CPS process, is extracted from a proof and executed.

In CPS models it is often implicitly expected that agents have knowledge about sequences of actions, whose executions lead agents to their goals, while the sequence construction process is not explicitly explained. Our CPS model is more planning-centric. First an agent is trying to find a plan that achieves its goals. Doing this, the agent may discover that either this is not possible or it is more efficient to involve other agents into problem solving process. However, since other agents may be self-interested, they may propose their offers and start a negotiation process. The process lasts until a (shared) plan has been found. The plan determines agents commitments and takes into account requirements determined during the negotiation.

In order to stimulate cooperation, agents should have a common goal [21]. We assume that all agents have a common meta-goal: as much agents as possible should become satisfied during run-time. All agents ask for minimum they need and provide maximum they can, during negotiation. This is biased with distributed theorem proving strategies. During negotiation the offers are derived using Partial Deduction (PD) for LL. PD allows determining missing links between proof fragments.

The rest of the paper is organised as follows. In Section 2 we present a general model of distributed problem solving and illustrate it with a working example. Section 3 gives an introduction to LL and PD. Section 4 proceeds with the working example from Section 2 by applying LL theorem proving and PD for CPS. Section 5 reviews related work and Section 6 concludes the article.

2 General CPS model and a working example

For further usage we define an agent $A_i$ as a triple

$$A_i = (I_i, S_i, G_i),$$

where:

- $I_i$ represents a set of agent’s capabilities in form $Delete \rightarrow Add$. $Delete$ and $Add$ are sets of formulae, which are respectively deleted from and added to the agent’s current state $S_i$, if the particular capability is applied. The capability can be applied only if $Delete \subseteq S_i$. 
- \( S_i \) is a set of literals representing the current state of the agent.
- \( G_i \) is a set of literals representing the goal state of the agent.

By agent’s capabilities we mean actions that an agent can perform. While performing a capability \( X \mapsto Y \), an agent consumes resources denoted by \( X \) and generates resources referred with \( Y \). The current state of an agent reflects agent’s perception of the current world together with agent’s internal state. The word “state” instead of “beliefs” is used herein to emphasise that our approach in planning-centric. Moreover, it expresses explicitly that our approach is not related much to BDI theory. We write \( S'_i \) and \( G'_i \) to denote modification of \( S_i \) and \( G_i \) respectively. Similarly we write \( S''_i \) and \( G''_i \) to denote modifications of \( S'_i \) and \( G'_i \), and so forth. While representing states and goals of agents we write \( A^n \) to specify that there are \( n \) instances of objects \( A \) in a particular state or a goal.

While planning, agents may discover that a plan, whose execution would lead them from state \( S_i \) to goal \( G_i \), could not be found. Then they determine subproblems (missing links), which cannot be solved by themselves. A missing link with respect to \( S_i \) and \( G_i \) is a pair \( (S'_i, G'_i) \), which is achieved by applying agent’s capabilities to its initial state \( S_i \) in forward chaining and to its goal \( G_i \) in backward chaining manner. Thus \( S_i \mapsto \ast S'_i \) and \( G_i \mapsto \ast G'_i \), where \( \mapsto \ast \) represents application of a capability from \( I_i \) and \( \mapsto \ast \) denotes that 0 or more capabilities are applied in sequence.

For illustrating detection of missing links let us consider the case where \( S_i = \{A\}, G_i = \{D\} \) and \( I_i = \{A \mapsto B, C \mapsto D\} \). Then possible missing links are \( (\{B\}, \{C\}), (\{B\}, \{D\}), (\{A\}, \{C\}) \) and \( (\{A\}, \{D\}) \). If, for example, agent \( A_1 \) decides that the most relevant missing link is \( (\{B\}, \{C\}) \), it sends a message with \( S'_i = \{B\}, G'_i = \{C\} \) to another agent \( A_2 \). Additionally \( A_1 \) may send a list of its capabilities \( I_1 \) (or a fragment of \( I_1 \) ) that might help \( A_2 \) in reasoning about possible offers.

Our general CPS model is depicted in Figure 1. Communication between agents goes via a Communication Adapter (CA), whose purpose is:

- to provide translations between agents’ communication languages and
- to change the viewpoint of proposals

Usage of Communication Adapter will be explained in details in Section 3.5. Our general CPS model can be described as follows:

1. Agent \( A_1 \) tries to generate a plan from elements of \( I_1 \), such that execution of the plan would lead \( A_1 \) from state \( S_i \) to \( G_i \). However, \( A_1 \) fails to construct such a solution.
2. Agent \( A_1 \) identifies possible missing links for achieving a complete solution. Every missing link derived from \( S_i \) and \( G_i \) is presented with a pair \( (S'_i, G'_i) \).
3. Agent \( A_1 \) delivers missing links to other agents and asks them to solve the missing links. If no answers are received, the problem is not solvable.
4. Agent \( A_2 \) agrees to help \( A_1 \).
5. If \( A_2 \) finds a complete solution for link \( (S'_i, G'_i) \), then \( S''_i \) and \( G''_i \) will be just copies of \( S'_i \) and \( G'_i \) respectively and \( P \) is a complete solution. Otherwise,
$S''_1$ and $G''_1$ are new missing links based on $S'_1$ and $G'_1$ and $P$ is a partial solution to the initial problem.

6. $A_2$ delivers $(S''_1, G''_1)$ and $P$ back to $A_1$.

7. Additionally, $A_2$ may require that $A_1$ helps to solve a missing link $(G''_2, S''_2)$ in return of a solution $P$ and a (new) missing link $(S''_1, G''_1)$. The link $(G''_2, S''_2)$ is achieved through application of CA to link $(S''_2, G''_2)$.

8. Agent $A_2$ may send the new missing link $(S''_1, G''_1)$ further to other agents as well for solving. In that case $A_2$ acts as a mediator agent.

Negotiation proceeds until the set of possible missing links will be exhausted or a solution satisfying both agents is found. As an extreme case all agents may be explicitly aware of each other capabilities, beliefs and goals a priori. In this case less messages are sent during negotiation, however, agents’ privacy, in terms of hiding their internal state, goal and capabilities, is less supported. In general, it is also possible that an agent is involved into negotiation with several agents simultaneously and at each step selects the best available offer. While implementing the presented model we apply LL theorem proving for planning and PD for determining subtasks, which have to be solved by other agents. The iterative CPS process is depicted in Figure 2.

Taking into account the presented general model, our working example is described as follows. Let us assume that two students, John and Peter, are looking for ways to relax after long days of studying and a final successful examination. John has a CD and he wants to listen music ($G_{John} = \{\text{Music}\}$). Unfortunately, his CD player is broken and this makes his goal non-realistic. John has also decided to visit a library to return books and this gives him possibility to return also books of other students when this may be useful for him. For covering all his expenses, related to relaxing, John has 10 USD. Considering that he has a broken CD player and a CD his initial state is as follows $S_{John} = \{\text{Dollar}^{10}, \text{CD}, \text{BrokenCDPlayer}\}$ and his capabilities are $\Gamma_{John} = \{\text{Books} \mapsto \text{BooksReturned}, \text{CDPlayer&CD} \mapsto \text{Music}\}$. 
Peter is skilled in electronics and can repair the CD player. He has decided to spend his day in a park with his girlfriend. However, he has to return books to the library. Since he has to take a taxi to reach the library, he has to spend 10 USD to cover his transportation expenses. This does not match well with his goals, since he has only 15 USD while he needs 25 USD for food, drinks and attractions in the park. Therefore he lacks 20 USD to achieve his goals. Peter’s initial state, goal and capabilities are described as follows: \( S_{Peter} = \{ Dollar^{15}, Books \} \), \( G_{Peter} = \{ BooksReturned, Beer \} \) and \( \Gamma_{Peter} = \{ Dollar^{10} & Books \mapsto BooksReturned, BrokenCDPlayer \mapsto CDPlayer, Dollar^{20} \mapsto Beer \} \).

Taking into account \( \Gamma_{John}, G_{John} \) and \( S_{John}, \) John constructs new state and goal and asks Peter whether he can repair his CD player: \( S'_{John} = \{ BrokenCDPlayer \} \), \( G'_{John} = \{ CDPlayer \} \). Peter decides to take advantage of the situation. He agrees to repair the CD player and asks 20 USD for performing this task: \( S'_{Peter} = \{ \} \), \( G'_{Peter} = \{ Dollar^{20} \} \), \( G''_{John} = G'_{John} \) and \( S''_{John} = S'_{John} \).

However, John has only 10 USD. Therefore he discloses additional information about his capabilities to Peter \( \Gamma''_{John} = \{ Books \mapsto BooksReturned \} \). Peter discovers that John has decided to visit a university library and agrees to decrease a fee for repairing the CD player by 10 USD, if John delivers his books to the library. John agrees and negotiation is successfully finished. During the negotiation agents’ plans have also been determined.

3 Formalisation of CPS process

3.1 Linear logic

LL is a refinement of classical logic introduced by J.-Y. Girard to provide means for keeping track of “resources”. In LL two assumptions of a propositional constant \( A \) are distinguished from a single assumption of \( A \). This does not apply in classical logic, since there the truth value of a fact does not depend on the number of copies of the fact. Indeed, LL is not about truth, it is about computation.

Although LL is not the first attempt to develop resource-oriented logics, it is by now the most investigated one. Since its introduction LL has enjoyed increasing attention both from proof theorists and computer scientists. Therefore,
because of its maturity, LL is useful as a declarative language and an inference kernel.

In following we are considering intuitionistic multiplicative additive fragment of LL (IMALL or MALL) consisting of multiplicative conjunction (⊗), additive disjunction (+), additive conjunction (&) and linear implication (→). In terms of resource acquisition the logical expression \( A \otimes B \vdash C \otimes D \) means that resources \( C \) and \( D \) are obtainable only if both \( A \) and \( B \) are obtainable. After the sequent has been applied, \( A \) and \( B \) are consumed and \( C \) and \( D \) are generated.

The expression \( A \vdash B \otimes C \) in contrary means that, if we have resource \( A \), we can obtain either \( B \) or \( C \), but we do not know which one of those. The expression \( A \& B \vdash C \) on the other hand means that while having resources \( A \) and \( B \) we can choose, which one of them to trade for \( C \). Therefore it is said that \( \otimes \) and \( \& \) are representing external and internal choice.

To illustrate preceding let us consider the following LL sequent from [14]—
\[(D \otimes D \otimes D \otimes D \otimes D) \vdash (H \otimes C \otimes (O&S)\otimes I) \otimes (P \otimes I)\],
which encodes a fixed price menu in a fast-food restaurant: for 5 dollars \( D \) you can get an hamburger \( H \), a coke \( C \), either onion soup \( O \) or salad \( S \) depending, which one you select, all the french fries \( F \) you can eat plus a pie \( P \) or an ice cream \( I \) depending on availability (restaurant owner selects for you). The formula \( !F \) here means that we can use or generate a resource \( F \) as much as we want—the amount of the resource is unbounded.

To increase the expressiveness of formulae, we are using in the following sometime abbreviation \( a^n = a \otimes \ldots \otimes a \), for \( n \geq 0 \).

Lincoln [15] summarises complexity results for several fragments of LL. Propositional MALL is indicated to be PSPACE-complete, whilst first-order MALL is at most NEXPTIME-hard. If we would discard additives \( \oplus \) and \( \& \) from MALL, we would get multiplicative LL (MLL). Both, propositional and first-order MLL, are NP-complete. It is also identified that these complexity results do not change, if intuitionistic fragments of LL are considered. These results hint that for practical computations either MLL or propositional MALL (or their intuitionistic variants MILL and MALL (IMALL), respectively) could be used. The complete set of IMALL inference rules is given in Appendix A.

3.2 Partial deduction and LL

Partial deduction (PD) (or partial evaluation of logic programs first introduced in [11]) is known as one of optimisation techniques in logic programming. Given a logic program, partial deduction derives a more specific program while preserving the meaning of the original program. Since the program is more specialised, it is usually more efficient than the original program, if executed. For instance, let \( A, B, C \) and \( D \) be propositional variables and \( A \rightarrow B, B \rightarrow C \) and \( C \rightarrow D \) computability statements in LL. Then possible partial deductions are \( A \rightarrow C, B \rightarrow D \) and \( A \rightarrow D \). It is easy to notice that the first corresponds to forward chaining (from beliefs to goals), the second to backward chaining (from goals to beliefs) and the third could be either forward or backward chaining.
Lloyd and Shepherdson [16] formalised PD for normal logic programs and showed its correctness with respect to Clark’s program completion semantics. Partial deduction in logic programming is often defined as unfolding of program clauses.

Although the original motivation behind PD was to deduce specialised logic programs with respect to a given goal, our motivation for PD is a bit different. We are applying PD for determining subtasks, which cannot be performed by a single agent, but still are possibly closer to a solution than an initial task. This means that given a state \( S \) and a goal \( G \) of an agent we compute a new state \( S' \) and a new goal \( G' \). This information is forwarded to another agent for further inference. Similar approach has been applied by Matskin and Komorowski [17] in automatic software synthesis. One of their motivations was debugging of declarative software specification.

The main problem with PD in LL is that although new derived states and goals are sound with respect to an initial specification, they may not preserve completeness anymore. This is due to resource-consciousness of LL—if a wrong proof branch is followed, initial beliefs may be consumed and thus further search become more limited. Therefore agents have to search, in the worst case, all possible PDs of initial specification to preserve completeness of distributed search mechanism. In [16] completeness and soundness issues of PD are considered for classical logic programs. Issues of complexity, completeness and soundness of PD in LL will be considered within another paper.

The following inference figures are applied as PD steps for back- and forward chaining:

Backward chaining step \( R_b(L_i) \):

\[
\begin{array}{c}
L \\
\top \\
\end{array}
\]

Forward chaining step \( R_f(L_i) \):

\[
\begin{array}{c}
L \\
\top \\
\end{array}
\]

\( L_i \) in previous inference figures is a labelling for a particular LL axiom representing agent’s capability. PD steps \( R_f(L_i) \) and \( R_b(L_i) \), respectively, apply clause \( L_i \) to move the initial state towards the goal state or vice versa. \( A, B \) and \( C \) are multiplicative conjunctions. In \( R_b(L_i) \) inference figure formulae \( B \otimes C \)
and \( A \otimes C \) denote respectively \( G \) and \( G' \). Thus the inference figure encodes that, if there is an extralogical axiom \( \vdash A \rightarrow B \), then we can change goal \( B \otimes C \) to \( A \otimes C \). Analogously, in the inference figure \( R_f(L_i) \) formulae \( B \otimes C \) and \( A \otimes C \) denote respectively \( S \) and \( S' \). And the inference figure encodes that, if there is an extralogical axiom \( \vdash B \rightarrow A \), then we can change initial state \( B \otimes C \) to \( A \otimes C \).

### 3.3 Agents in LL

An agent is presented with the following LL sequent:

\[ \Gamma; S \vdash G, \]

where \( \Gamma \) is a set of extralogical LL axioms representing agent’s capabilities, \( S \) is the initial state and \( G \) the goal state of an agent. Both \( S \) and \( G \) are multiplicative conjunctions of literals. Every element of \( \Gamma \) is in form

\[ \vdash I \rightarrow O, \]

whereas \( I \) and \( O \) are multiplicative conjunctions of formulae, which are respectively consumed and generated when a particular capability is applied. It has to be noted that a capability can be applied only, if conjunctions in \( I \) form a subset of conjuncts in \( S \).

### 3.4 Encoding offers in LL

Harland and Winikoff [8] presented the first ideas about applying LL theorem proving for agent negotiation. The main advantages of LL over classical logic is its resource-consciousness and existence two kinds of nondeterminism. Both internal and external nondeterminism in negotiation rules can be represented. In the case of internal nondeterminism a choice is made by resource provider, whereas in the case of external nondeterminism a choice is made by resource consumer. For instance, formula \( \text{Dollar}^5 \rightarrow \text{Beer} \oplus \text{Soda} \) means that an agent can provide either some \( \text{Beer} \) or \( \text{Soda} \) in return for 5 dollars, but the choice is made by the provider agent. The consumer agent has to be ready to obtain either a beer or a soda. The formula \( \text{Dollar} \rightarrow \text{Tobacco} & \text{Lighter} \) in contrary means that the consumer may select which resource, Tobacco or Lighter, s/he gets for a Dollar.

In the context of negotiations, operators \& and \( \oplus \) have symmetrical meanings—what is \( A \oplus B \) for one agent, is \( A & B \) to her/his partner. It means that if one agent gives to another an opportunity to choose between \( A \) and \( B \), then the former agent has to be ready for providing both choices, \( A \) and \( B \). When initial resources owned by agents and expected negotiation results have been specified, LL theorem proving is used for determining the negotiation process.

We augment the ideas of Harland and Winikoff by allowing trading also services (agent capabilities) for resources and vice versa. This is a step further to the world where agents not only exchange resources, but also work for other agents in order to achieve their own goals. We write \( A \vdash B \rightarrow C \) to indicate that an agent can trade resource \( A \) for a service \( B \rightarrow C \).
3.5 Communication Adapter

In [18] bridge rules are used for translating formulae from one logic into another, if agents exchange offers. We adopt this idea for Communication Adapter (CA) for two reasons. First, it would allow us to encapsulate agents’ internal state and second, while offers are delivered from one agent to another, viewpoint to the offer is changing and internal and external choices are reversed. By viewpoint we mean agent’s role, which can be either receiver or sender of an offer.

The CA rule is described in the following way. As long as formulae on the left and the right hand side of sequents consist of only $\otimes$ and $\neg\otimes$ operators, the left and the right hand sides of sequents are inversed. However, if formulae contain disjunctions, their types have to be inversed. This has to be done because there are 2 disjunctions in LL—one with internal and another with external choice. Since internal and external choices are context-dependent, they have to be inversed, if changing viewpoints. For instance, sequent $A \otimes (A \rightarrow B) \vdash C \oplus D$ is translated to $C \& D \vdash A \otimes (A \rightarrow B)$ through the CA rule:

$$
\frac{B_j \vdash A_i}{i} \quad \frac{A_i \vdash B_j}{j} \quad \text{CA}
$$

In the CA rule $A$ and $B$ consist of multiplicative conjunctions and linear implications. We allow $\&$ and $\oplus$ only in the left and the right hand side of a sequent, respectively. Due to LL rules $R\&$ and $L\oplus$ the following conversions are allowed:

$$
D \vdash \bigwedge_j D_j \implies \bigcup_j (D \vdash D_j)
$$

$$
\bigoplus_j D_j \vdash D \implies \bigcup_j (D_j \vdash D)
$$

Therefore we do not lose anything in expressive power of LL, when limiting where disjunctions may occur, but proposals are kept declaratively more understandable.

Although our bridge rule is intended for agents reasoning in LL only, additional bridge rules may be constructed for communication with other non-LL agents. However, it should be mentioned that there is no one-to-one translation between most of logics and therefore information loss may occur during translation.

4 Running example

Let us consider encoding of the working example from Section 2 in LL and explain how to represent agents’ capabilities (actions they can perform), negotiation arguments, agents’ states and goals with LL formulae. In order to keep the
resulting proof simple, we take advantage of propositional MILL (a fragment of MAILL) only.

The initial scenario is described formally as follows:

\[ \Gamma_{\text{John}} = \vdash_{\text{John}} \text{Books} \rightarrow_{\text{returnBooks}} \text{BooksReturned} \]
\[ \vdash_{\text{John}} \text{CDPlayer} \odot \text{CD} \rightarrow_{\text{playMusic}} \text{Music} \]
\[ \vdash_{\text{Peter}} \text{Dollar}^{10} \odot \text{Books} \rightarrow_{\text{returnBooks}} \text{BooksReturned} \]
\[ \Gamma_{\text{Peter}} = \vdash_{\text{Peter}} \text{BrokenCDPlayer} \rightarrow_{\text{repairCDPlayer}} \text{CDPlayer} \]
\[ \vdash_{\text{Peter}} \text{Dollar}^{55} \rightarrow_{\text{buyBeer}} \text{Beer} \]

The sets of extralogical axioms \( \Gamma_{\text{John}} \) and \( \Gamma_{\text{Peter}} \) represent capabilities of John and Peter, respectively. We write \( \vdash_X \) to indicate that a capability is provided by \( X \), \( \neg_Y \) labels a capability with name \( Y \). The internal state of John is described by the following sequent:

\[ \Gamma_{\text{John}}; \text{Dollar}^{10} \odot \text{CD} \odot \text{BrokenCDPlayer} \vdash_{\text{John}} \text{Music} \]

This means that John has 10 USD, a CD and a broken CD player. His goal is to listen music. Peter’s state and goal are described by another sequent:

\[ \Gamma_{\text{Peter}}; \text{Dollar}^{15} \odot \text{Books} \vdash_{\text{Peter}} \text{BooksReturned} \odot \text{Beer} \]

We write \( B, BR, BE, CD, P, BP, M \) and \( D \) to denote \( \text{Books}, \text{BooksReturned}, \text{Beer}, \text{CD}, \text{CDPlayer}, \text{BrokenCDPlayer}, \text{Music} \) and \( \text{Dollar} \) respectively. Given John’s and Peter’s capabilities and internal states, both agents start individually with theorem proving. Initially they fail, since they are unable to reach their goals individually. Then PD in LL is applied to the same set of formulae and new subtasks are derived. These subtasks indicate problems, which could not be solved by agents themselves and need cooperation with other agents. In particular, John has to ask help for solving the following sequent, which is derived by PD:

\[ \text{D}^{10} \vdash_{\text{John}} \text{BP} \rightarrow \text{P} \]

The sequent is achieved by applying the backward chaining step of PD in the following way (to allow shorter proof, we write here \( \vdash \) instead of \( \vdash_{\text{John}} \)):

\[
\frac{\text{CD} \vdash \text{CD}} {\text{CD} \vdash \text{CD}} \quad \text{Id}
\frac{\text{D}^{10} \vdash_{\text{John}} \text{BP} \rightarrow \text{P}} {\text{D}^{10} \odot \text{BP} \vdash \text{P}} \quad \text{Shift}
\frac{\text{CD}, \text{D}^{10} \odot \text{BP} \vdash \text{CD} \odot \text{P}} {\text{D}^{10} \odot \text{BP} \odot \text{CD} \vdash \text{CD} \odot \text{P}} \quad \text{R} \odot
\frac{\text{D}^{10} \odot \text{BP} \odot \text{CD} \vdash \text{John} \text{M}} {\text{D}^{10} \odot \text{BP} \odot \text{CD} \vdash \text{John} \text{M}} \quad \text{R}_b(\text{playMusic})
\]

where \( \text{Shift} \) is another inference figure:

\[
\frac{\text{C} \vdash \text{A} \rightarrow \text{B}} {\text{A} \vdash \text{A} \rightarrow \text{B} \vdash \text{B}} \quad \text{Id}
\frac{\text{A} \vdash \text{A} \rightarrow \text{B} \vdash \text{B}} {\text{C} \odot \text{A} \vdash \text{B}} \quad \text{Cut}
\]
The purpose of the inference figure above, is to transform sequents of the form $C \otimes A \vdash B$ to the form $C \vdash A \rightarrow_{\omega} B$, which is our representation sequent for agents’ capabilities. It has to be mentioned that $C$ is possibly an empty formula.

Peter sees the offer as follows:

$$BP \rightarrow_{\omega} P \vdash_{Peter} D^{10}$$

Changing a viewpoint for offers is done by Communication Adapter (CA, see Section 3.5).

Since Peter can repair the CD player, he agrees partially with the proposal. However, because he needs 20 USD instead of the proposed 10 USD, he increases the number of dollars required. In addition, assuming that John may not have the asked amount of money, Peter provides him with information about tasks ($\Gamma_{Peter}$), which may be traded for money:

$$\Gamma_{Peter}; BP \rightarrow_{\omega} P \vdash_{Peter} D^{20}$$

This is viewed by John as follows:

$$\Gamma_{Peter}; D^{20} \vdash_{John} BP \rightarrow_{\omega} P$$

By exploring $\Gamma_{Peter}$ John discovers that both his and John’s goals would be reached, if he delivers Peters’ books to the library for 10 USD and Peter repairs the CD player for 20 USD. Peter accepts the offer.

Another possible solution could be that Peter repairs the CD player for 10 USD and John delivers his books. However, while the former solution makes it possible to introduce binding contracts between agents, the latter expects that both agents trust each other. Depending on the proof strategy, we can represent different solutions to cooperation problem solving.

The final proposal from John is presented as the following sequent:

$$\Gamma_{Peter}; D^{20} \otimes (D^{10} \otimes B \rightarrow BR) \vdash_{John} BP \rightarrow_{\omega} P$$

After applying CA it is perceived by Peter as follows:

$$\Gamma_{Peter}; BP \rightarrow_{\omega} P \vdash_{Peter} D^{20} \otimes (D^{10} \otimes B \rightarrow BR)$$

During communication agents can send information about their capabilities ($\Gamma_{Peter}$ in the current case). This provides their partners with hints about possible counteroffers. The following proof is constructed for John during the above-described series of message exchange (we omit $\Gamma_{John}$ and $\Gamma_{Peter}$ in the proof for simplicity):
John’s plan can be extracted from the proof and then executed. In the proof action `returnBooks` occurs twice. While `returnBooks` means that John has to execute it, `returnBooks`Peter explicitly represents that Peter has to be charged with 10 dollars for performing previous action. A proof for Peter can be constructed in a similar way.

The proof above does not present the whole negotiation process. It rather demonstrates the result of negotiation and team and plan formation. The proof also indicates how John and Peter have joined the efforts to achieve their goals. The dependency of plans for John and Peter is shown in Figure 3.

![Diagram](image)

**Fig. 3.** Interdependent plans.

Arrows in Figure 3 indicate a partial order of actions to be executed (or capabilities applied) and the line separates actions performed by different agents (actions above the line are executed by John and actions below the line are executed by Peter).

## 5 Related work

As it has been indicated in [10] negotiation is the most fundamental and powerful mechanism for managing inter-agent dependencies at run-time. Negotiation may be required both for self-interested and cooperative agents. It allows to reach a mutually acceptable agreement on some matter by a group of agents.

Kraus et al [12] give a logical description for negotiation via argumentation for BDI agents. They classify arguments as threats and promises, which are identified as most common arguments in human negotiations. In our case only promises are considered, since in order to figure out possible threats to goals of particular agents, agents’ beliefs, goals and capabilities should be known in
advance to the persuader. We assume, that our agents do not explicitly communicate about their internal state.

Fisher [4] introduced the idea of distributed theorem proving in classical logic as agent negotiation. In his approach all agents share the common view to the world and if a new clause is inferred, all agents would sense it. Inferred clauses are distributed among agents via broadcasting. Then, considering the received information, agents infer new clauses and broadcast them further again. Although agents have a common knowledge about inferred clauses, they may have different sets of inference rules. Distribution of a collection of rules between agents means that different agents may have different capabilities and make different inferences. The latter implies that different agents contribute to different phases of proof search.

Parsons et al [18] defined negotiation as interleaved formal reasoning and arguing. Arguments and counterarguments are derived using theorem proving while taking into consideration agents’ own goals. While Parsons et al [18] perform reasoning in classical logic, it is possible to infer missing clauses needed for achieving a goal. The situation gets more complicated, if several instances of a formulae are available and moreover, the actions performed by agents or resources they spend can be interdependent. Therefore, inference in LL is not so straightforward, since some clauses are “consumed” while inferring others. Due to the aforementioned reasons we apply PD to determine missing parts of a proof. Then a missing part is announced to other possibly interested agents.

Case-based planning has been used for coordinating agent teams in [5]. The planner generates a so called shared mental model of the team plan. Then all agents adapt their plans to the team plan. This work is influenced by the joint intentions [13,3] and shared plans [7] theory.

In [19] agent coordination is performed through task agents by planning. First problem solving goals are raised and then solutions satisfying these goals are computed. Finally these plans are decomposed and coordinated with appropriate task or other agents for plan execution, monitoring and result collection. Other agents are information and interface agents, for information collection and interfacing with a human user, respectively.

The joint intentions theory [3] determines the means how agents should act to fulfill joint goals, when they should exchange messages, synchronize between themselves, leave the team, etc. It also determines when the joint goal is considered to be achieved or when and how to break up commitment to it, if it should, for instance, turn out that one agent is not able anymore to perform its task(s). Decision making about whether a goal has been achieved, is not achievable or there is no need to achieve it anymore, is based on consensus—every agent can initiate a discussion through which consensus is (presumably) achieved. Then everybody acts as stated by the consensus. However, it is not stated how joint goals are formed through negotiation or other processes.

One of the first formalisations of cooperative problem solving is given by Wooldridge and Jennings [21] (also other approaches presented so far are reviewed there). One of the earliest implemented general models of teamwork is
described in [20], which is based of joint intentions theory and partially borrows from shared plans theory.

In [1] LL has been used for prototyping multi-agent systems at conceptual level. Because of fixed semantics of LL, it is possible to verify whether a system functions at conceptual level as intended. Although the prototype LL program is executable, it is still too high level to produce a final agent-based software. Thus another logic programming language is embedded to compose the final software.

Harland and Winikoff [9] address the question how to integrate both proactive and reactive properties of agents into LL programming framework. Here forward chaining is used to model the reactive behaviour of an agent and backward chaining for the proactive behaviour respectively. That sort of computation is called there mixed mode computation, because both, forward and backward chaining are allowed.

6 Conclusions

In contrast to other formalisations of CPS, we presented a computational model, where the main emphasis is placed on negotiation and planning. Need for cooperation is sensed, if an agent alone is unable to find a plan for achieving its goals. Agent teams are formed on basis of interdependent plans.

For formalising our general CPS model we chose LL. Planning is implemented on top of LL theorem proving. Initially a planning problem is coded in terms of LL sequents and theorem proving is applied for generating constructive proofs. From generated proofs then plans are extracted. We specified negotiation as distributed LL theorem, whereas PD is used for generating offers. Offers are generally in form—I can grant you X if you provide me with Y, where X and Y can be both resources and capabilities of agents. X can be empty.

We have implemented a planner on top of theorem prover for first-order MILL and performed initial experiments. The planner is available at address http://www.idi.ntnu.no/~peep/RAPS.

Acknowledgements

This work is partially supported by the Norwegian Research Foundation in the framework of the Information and Communication Technology (IKT-2010) program—the ADIS project. We would like to thank the anonymous referees for their useful comments.

References

A IMALL rules

Logical axiom and Cut rule:

\[ \frac{\Gamma \vdash A, \Delta \quad \Gamma', A \vdash \Delta'}{\Gamma; \Gamma' \vdash \Delta, \Delta'} \quad \text{(Cut)} \]

\[ A \vdash A \quad \text{(Axiom)} \]

Rules for the propositional constants:

\[ \vdash 1 \quad \frac{\Gamma \vdash A}{\Gamma; 1 \vdash A} \]

\[ \frac{\Gamma, A, B \vdash \Delta}{\Gamma; A \otimes B \vdash \Delta} \quad \frac{\Gamma, A, \Delta \quad \Gamma' \vdash B, \Delta'}{\Gamma; \Gamma' \vdash A \otimes B, \Delta, \Delta''} \quad \text{(R\otimes)} \]

\[ \frac{\Sigma_1 \vdash A,B, \Sigma_2 \vdash C}{\Sigma_1, (A \rightarrow B), \Sigma_2 \vdash C} \quad \frac{\Sigma, A \vdash B}{\Sigma \vdash (A \rightarrow B)} \quad \text{(R\rightarrow)} \]

\[ \frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \oplus B \vdash \Delta} \quad \frac{\Gamma \vdash A, \Delta}{\Gamma; A \oplus B, \Delta} \quad \frac{\Gamma \vdash A, \Delta}{\Gamma; A \oplus B, \Delta} \quad \text{(R\oplus)} \]

\[ \frac{\Gamma, A \vdash \Delta}{\Gamma; A \& B \vdash \Delta} \quad \frac{\Gamma \vdash A, \Delta}{\Gamma; A \& B \vdash \Delta} \quad \frac{\Gamma \vdash A, \Delta}{\Gamma; A \& B, \Delta} \quad \text{(R&)} \]

Rules for quantifiers:

\[ \frac{\Gamma, A[a/x] \vdash \Delta}{\Gamma; \forall x A \vdash \Delta} \quad \frac{\Gamma, \Delta, A[t/x]}{\Gamma \vdash \Delta, \forall x A} \quad \text{(L\forall)} \]

\[ \frac{\Gamma, A[t/x] \vdash \Delta}{\Gamma; \exists x A \vdash \Delta} \quad \frac{\Gamma \vdash A[a/x], \Delta}{\Gamma \vdash \exists x A, \Delta} \quad \text{(L\exists)} \]

where \( t \) is not free in \( \Gamma \) and \( \Delta \).