

Symbolic Negotiation Revisited

Peep K ngas

Norwegian University of Science and Technology
Department of Computer and Information
Science
Trondheim, Norway
peep@idi.ntnu.no

Mihhail Matskin

Royal Institute of Technology
Department of Microelectronics
and Information Technology
Kista, Sweden
misha@imit.kth.se

ABSTRACT

In this paper we propose a formalism for symbolic negotiation. We regard symbolic negotiation as cooperative problem solving (CPS), which is based on symbolic reasoning and is extended with negotiation-specific rules. The underlying CPS formalism was previously presented in [3]. Here we extend the results and position symbolic negotiation according to other distributed problem solving mechanisms.

Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]: Multiagent systems; I.2.3 [Deduction and Theorem Proving]: Deduction; F.4.1 [Mathematical Logic]: Computational logic

General Terms

Algorithms, Theory

Keywords

Symbolic reasoning, partial deduction, cooperative problem solving, negotiation

1. INTRODUCTION

Symbolic negotiation is regarded in the field of computer science as a process, where parties try to reach an agreement on the high-level means for achieving their goals by applying symbolic reasoning techniques. Formalisation of symbolic negotiation could contribute both to the studies of human behaviour and multi-agent systems. In the former field the rational part of human reasoning could be modelled, while in the latter field self-organising systems with adaptive behaviour could be implemented.

Previously, K ngas and Matskin proposed [2] that symbolic negotiation could be formalised as partial deduction (PD) for linear logic (LL). In [3] a formalisation of PD for LL was presented and its soundness and completeness was

proved. The paper regarded symbolic negotiation as interleaved cooperative problem solving (CPS) and plan modification. However, the paper formalised only the CPS part of the process. In this paper we extend the proposed CPS formalism with plan modification operators and then analyse symbolic negotiation as a whole. We also formalise the coalition formation process and analyse its effect to CPS and symbolic negotiation. Anyway, due to the limited space we do not present proofs for propositions and theorems in this paper.

We would like to underline that from a computational point of view, we can regard CPS as AI planning and symbolic negotiation as plan reuse/repair. It has been shown [5] that from problem solving point of view in general neither planning from scratch nor plan repair has an advantage over each-other. Therefore we expect both CPS and symbolic negotiation to be computationally equivalent. Moreover, both CPS and symbolic negotiation lead to the same results as we prove in this paper.

However, compared to CPS, symbolic negotiation provides a more human-like way of problem solving, which can be more naturally followed by human participants. In addition, symbolic negotiation may encode a sort of search heuristics, which would make CPS computationally less demanding. These heuristics, however, are not discussed in this paper.

2. LINEAR LOGIC AND PARTIAL DEDUCTION

2.1 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.

We consider the !-Horn fragment of LL (HLL) [1] consisting of multiplicative conjunction ($\otimes$), linear implication ($\multimap$) and “of course” operator ($!$). 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 produced.

While implication $A \multimap B$ as a computability statement clause in HLL could be applied only once, $!(A \multimap B)$ may be used an unbounded number of times. When $A \multimap B$ is

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS’06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...\$5.00.

applied, then literal A becomes deleted from and B inserted to the current set of literals. If there is no literal A available, then the clause cannot be applied. In HLL $!$ cannot be applied to formulae other than linear implications.

In order to explain HLL and demonstrate some other features of LL, not presented in HLL, let us consider the following LL sequent from [4]— $(D \otimes D \otimes D \otimes D) \vdash (H \otimes C \otimes (O \& S) \otimes !F \otimes (P \oplus I))$, 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.

2.2 Partial deduction

Here we describe basic definitions required to understand PD. Most of the definitions in this section are from [3], where they were used to define a CPS formalism. The presented rules have to read in bottom-up manner.

DEFINITION 1. *Resultant is a linear implication*

$$\vdash I \multimap_{\lambda a_1, \dots, a_n. f} O, n \geq 0,$$

where f is a term representing a function, which generates O from I by applying potentially composite functions over $a_1, \dots, a_n$.

DEFINITION 2. *Forward chaining PD step $\mathcal{R}_f(L_i)$ is defined as a rule*

$$\frac{B \otimes C \vdash G}{A \otimes C \vdash G} \mathcal{R}_f(L_i)$$

where L_i is a labeling of linear implication $\vdash A \multimap_{L_i} B$. A, B, C and G are multiplicative conjunctions.

DEFINITION 3. *Backward chaining PD step $\mathcal{R}_b(L_i)$ is defined as a rule*

$$\frac{S \vdash A \otimes C}{S \vdash B \otimes C} \mathcal{R}_b(L_i)$$

where L_i is a labeling of linear implication $\vdash A \multimap_{L_i} B$. A, B, C and S are multiplicative conjunctions.

PD steps $\mathcal{R}_f(L_i)$ and $\mathcal{R}_b(L_i)$, respectively, apply L_i in forward and backward chaining manner.

3. AGENTS AND COALITION FORMATION

3.1 Agent representation

An agent is presented with the following HLL sequent:

$$\Gamma; S \vdash G,$$

where Γ is a set of agent capabilities, S is the initial state and G is the goal state of the agent. Both S and G are multiplicative conjunctions of HLL formulae.

Every element of Γ is a linear implication $\vdash I \multimap O$, where I and O are formulae which are, respectively, consumed and generated when a particular capability is applied. It has to be mentioned that a capability can be applied only, if conjuncts in I form a subset of conjuncts in S . It should be

also underlined that in order to achieve their goals, agents have to construct (and then execute) the following plan from the elements of Γ :

$$\vdash S \multimap G.$$

Each agent is assumed to have a single task only. Whenever in an application an agent is supposed to have several task, it would be modelled in our framework as several agents, each having a single task.

DEFINITION 4. *Agent's task is a resultant in PD.*

Modifications of an agent's task are derived through PD. We consider all possible derivations of a task as instances of the same task. Moreover, we constrain an agent to consider only one derivation at time during distributed CPS. This would ensure that instances of the same task would not be solved concurrently.

DEFINITION 5. *Resource is a formula of HLL.*

3.2 Agent coalitions

The purpose of agent coalition formation in our formalism is to merge several tasks into a larger one. Then agents become able to perform tasks, which they would not have been able to perform individually. Thereby coalition formation could be seen as a way to optimize problem solving globally.

DEFINITION 6. *Agent coalition $\mathcal{C}$ is a structure $\langle \mathcal{A}, \mathcal{T}, \mathcal{R}, \triangleright, \sqcap \rangle$, where $\mathcal{A}$ is a set of participating agents, $\mathcal{T}$ is a coalition task and $\mathcal{R}$ is a set of resources shared between agents. $\triangleright$ is a mapping from agents to resources ($\triangleright : \mathcal{A} \times \mathcal{A} \rightarrow \mathcal{R}^*$), which one agent would deliver to another if a solution for a coalition task is found. Finally, $\sqcap$ is a mapping from a coalition task to agents in the coalition ($\sqcap : \mathcal{T} \rightarrow \mathcal{A}^+$), which agreed to solve the task together by sharing their resources.*

We interpret a coalition as an agreement between agents to solve their tasks together. In a coalition agents can also exchange resources directly between themselves. Additionally it should be emphasised that we model tasks as (composite) actions consuming and producing resources. Coalitions are formed by applying rule *merge*.

DEFINITION 7 (COALITION FORMATION RULE). *Coalition formation rule merge is defined as follows:*

$$\frac{\vdash A \otimes B \multimap C \otimes D}{\vdash A \multimap C \quad \vdash B \multimap D} \text{ merge}$$

Rule *merge* combines 2 tasks X and Y to create a task with the intended meaning: “task X could be performed by one agent, if another agent performs task Y ” and vice versa. From a distributed theorem proving point of view it means that tasks X and Y are combined together and their search spaces are merged. If a coalition includes all agents of a system, then problem solving is seen as solving a single task in a distributed manner. This contrasts with our general case, where multiple tasks are solved concurrently in a distributed way.

PROPOSITION 1. *Solution of a coalition task $\mathcal{T}$, which was achieved through PD, solves the tasks of all coalition members $[T]$.*

DEFINITION 8 (SOUNDNESS OF COAL. FORMATION). *If a solution of coalition task T solves the tasks of all participating agents $[T]$, then coalition formation is sound.*

PROPOSITION 2. *Coalition formation is sound with respect to PD.*

Coalition formation could be seen from problem solving point of view as merging several tasks into one. Thus instead of solving multiple tasks concurrently, with coalitions we would solve a composed task.

PROPOSITION 3. *If a set of tasks included in a coalition task is solvable through PD independently, then so is the composed coalition task.*

DEFINITION 9 (COMPLETENESS OF COAL. FORMATION). *If agents' tasks were solvable individually and their coalition task is solvable as well, then coalition formation is complete.*

PROPOSITION 4. *Coalition formation is complete with respect to PD.*

DEFINITION 10. *A coalition task is implicitly solved, if no PD steps are required to solve the coalition task.*

4. SYMBOLIC NEGOTIATION

In this section we define the symbolic negotiation process with respect to PD. We define also symbolic negotiation rules for operating over forward (Θ_f) and backward chaining (Θ_b). Additionally some other negotiation-specific definitions are given. Finally, we prove that symbolic negotiation is sound and complete.

DEFINITION 11. *Negotiation rule $\Theta_f(a_k/a_i)$ is defined as the following operational rule:*

$$\frac{\frac{\vdash (((I \setminus O(a_i)) \cup I(a_i)) \setminus I(a_k)) \cup O(a_k))}{\neg_{a_1, \dots, a_{i-1}, a_k, a_{i+1}, \dots, a_n} O}}{\frac{\vdash I \neg_{a_1, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_n} O}{I(a_k) \subseteq I(a_i) \quad O(a_i) \subseteq O(a_k)}} \Theta_f(a_k/a_i)$$

DEFINITION 12. *Negotiation rule $\Theta_b(a_k/a_i)$ is defined as the following operational rule:*

$$\frac{\frac{\vdash I \neg_{a_1, \dots, a_{i-1}, a_k, a_{i+1}, \dots, a_n} O}{(((O \setminus I(a_i)) \cup O(a_i)) \setminus O(a_k)) \cup I(a_k)}}{\frac{\vdash I \neg_{a_1, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_n} O}{I(a_k) \subseteq I(a_i) \quad O(a_i) \subseteq O(a_k)}} \Theta_b(a_k/a_i)$$

$I(a)$ and $O(a)$ in the preceding rules represent respectively resources required to apply and achieved by applying capability a . I represents resources available for solving a task while O represents resources, which must be achieved by the task. The preceding negotiation rules replace one PD step application in a task derivation with another. Rule $\Theta_f(a_k/a_i)$ replaces forward chaining steps while $\Theta_b(a_k/a_i)$ replaces backward chaining steps. The symbolic negotiation rules replace a capability in the derivation with another capability which consumed the same amount or less and generated the same amount or more resources than the previous capability. Therefore negotiation rules allow to optimise a task of agent X by agent Y . This means that some PD steps of agent X in its task are substituted by Y , if associated capabilities are more efficient in terms of resource consumption and generation.

DEFINITION 13 (SYMBOLIC NEGOTIATION). *Symbolic negotiation is PD with coalition formation and symbolic negotiation rules.*

DEFINITION 14 (CPS). *Cooperative problem solving is PD with coalition formation.*

DEFINITION 15 (COMPUTATIONAL EQUIVALENCE). *Two processes are computationally equivalent with respect to PD, if they would construct the same set of resultants.*

THEOREM 1. *CPS and symbolic negotiation are computationally equivalent.*

According to Theorem 1 symbolic negotiation could be view as a special case of CPS, where PD is guided at operational level.

DEFINITION 16. *Symbolic negotiation is sound, if it is sound with respect to PD, and all resultants achieved through symbolic negotiation are achievable through CPS.*

DEFINITION 17. *Symbolic negotiation is complete, if it is complete with respect to PD, and all resultants achieved through CPS are achievable through symbolic negotiation.*

THEOREM 2 (SOUNDNESS OF SYMBOLIC NEGOTIATION). *Symbolic negotiation is sound.*

THEOREM 3 (COMPLETENESS OF SYMB. NEGOTIATION). *Symbolic negotiation is complete.*

Soundness and completeness of our CPS formalism are directly derived from Proposition 2, Proposition 4 and completeness/soundness results of PD presented in [3].

Acknowledgements

This work was partially supported by the Norwegian Research Foundation in the framework of Information and Communication Technology (IKT-2010) program—the ADIS project.

5. REFERENCES

- [1] M. I. Kanovich. Linear logic as a logic of computations. *Annals of Pure and Applied Logic*, 67:183–212, 1994.
- [2] P. Küngas and M. Matskin. Linear logic, partial deduction and cooperative problem solving. In *Proceedings of the First International Workshop on Declarative Agent Languages and Technologies (in conjunction with AAMAS 2003), DALT'2003, Melbourne, Australia, July 15, 2003*, volume 2990 of *Lecture Notes in Artificial Intelligence*, pages 263–279. Springer-Verlag, 2004.
- [3] P. Küngas and M. Matskin. Partial deduction for linear logic—the symbolic negotiation perspective. In *Proceedings of the Second International Workshop on Declarative Agent Languages and Technologies (in conjunction with AAMAS 2004), DALT'2004, New York, USA, July 19, 2004*, volume 3476 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2005.
- [4] P. Lincoln. Linear logic. *ACM SIGACT Notices*, 23(2):29–37, 1992.
- [5] B. Nebel and J. Koehler. Plan reuse versus plan generation: A theoretical and empirical analysis. *Artificial Intelligence*, 76:427–454, 1995.