

Partial Deduction for Linear Logic—The Symbolic Negotiation Perspective

Peep Kungas, Mihhail Matskin★

{peep,misha}@idi.ntnu.no.

Norwegian University of Science and Technology, Trondheim, Norway

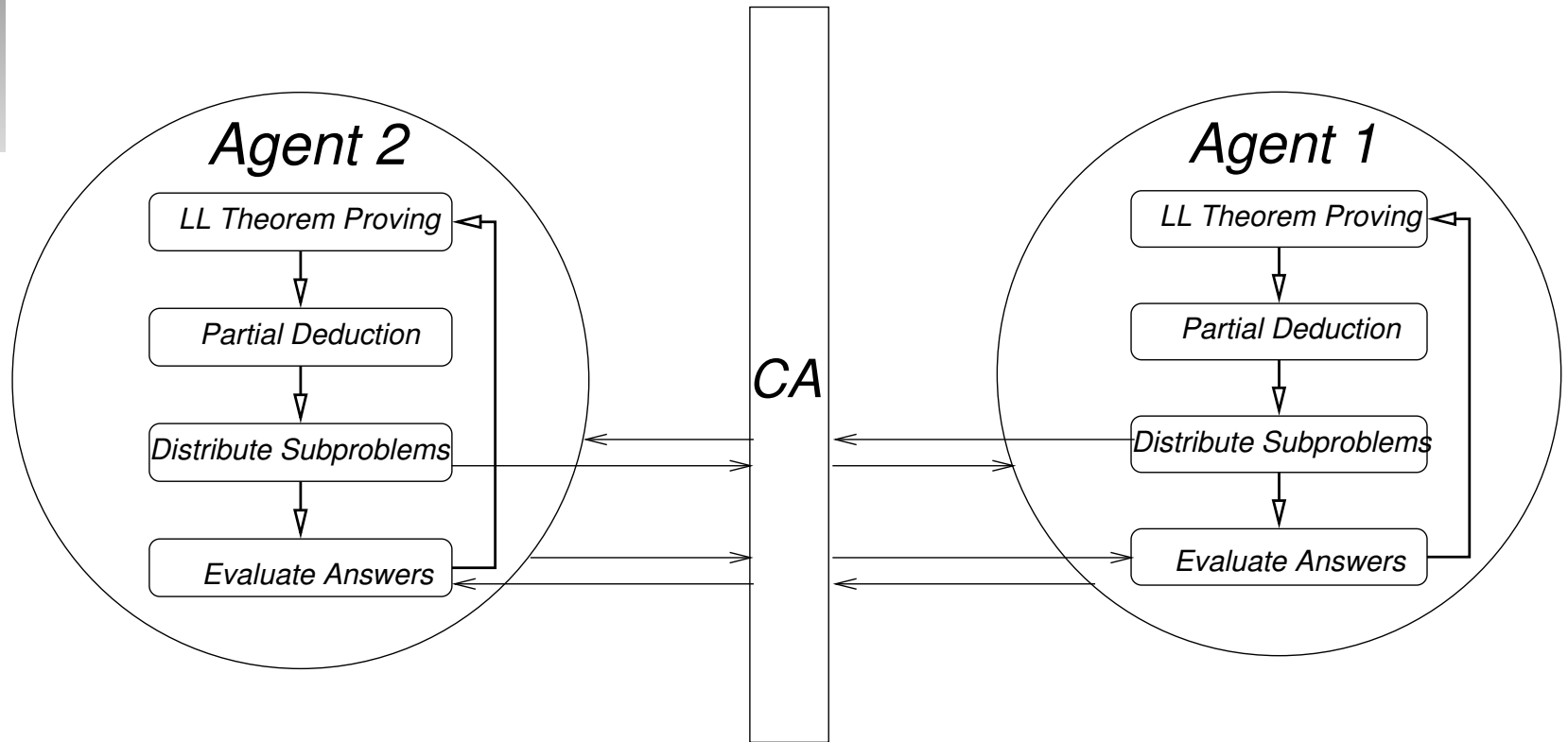
★ Royal Institute of Technology, Kista, Sweden

- Introduction and motivation
- Linear logic (LL)
- Partial deduction for LL
- Symbolic negotiation
- Partial deduction strategies
- Conclusions and future work

- *Formalisation and automation* of agent-related computational processes and interaction
- *Adaptive specification* of computational problems

Every process, which can be formalised, can be automated!

Architecture of Symbolic Negotiation



- The process of symbolic negotiation has been defined (outside computer science disciplines) as a process in which competing propositions attempt to establish claims to be worth believing
- In computer science symbolic negotiation is defined as negotiation through symbolic reasoning
- Formally, we define symbolic negotiation as PD for LL and some rules for information exchange

$(D \otimes D \otimes D \otimes D \otimes D) \vdash (H \otimes C \otimes !F \otimes (P \oplus I) \otimes (T \& E))$,
where

- D represents a dollar,
- H an hamburger,
- C a coke,
- F all the french fries you can eat,
- P a pie,
- I ice cream,
- T tea,
- and E espresso.

- A method of deducing new formulae step-by-step
- PD was initially meant for logic program optimisation
- Generally, PD = deduction + strategies
- Side-effect of PD in our framework is a partial plan

Benefits:

- Goal does not have to be specified completely
- Sometimes we cannot achieve the exact goal, but still may get something close to it

Forward and backward chaining PD steps $\mathcal{R}_f(L_i)$ and $\mathcal{R}_b(L_i)$:

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

where L_i is a labelling of $\text{CSC} \vdash A \multimap_{L_i} B$.

A, B, C, G and S are multiplicative conjunctions.

$$\frac{!A \otimes !A \otimes B \vdash C}{!A \otimes B \vdash C} \mathcal{R}_{C_l}$$

$$\frac{A \otimes B \vdash C}{!A \otimes B \vdash C} \mathcal{R}_{L_l}$$

$$\frac{B \vdash C}{!A \otimes B \vdash C} \mathcal{R}_{\dots}$$

$$\frac{!A \otimes A^n \otimes B \vdash C}{!A \otimes B \vdash C} \mathcal{R}_{!_l}(n)$$

where A is a literal, while B and C are multiplicative conjunctions. $A^n = \underbrace{A \otimes \dots \otimes A}_n$, for $n > 0$.

Traveller \mathcal{T} :

$From \otimes To \vdash Booking,$

$$\Gamma_{\mathcal{T}} = \begin{array}{l} \vdash From \otimes To \multimap_{findSchedule} Schedule, \\ \vdash Site \multimap_{getPassword} Password. \end{array}$$

Flight company \mathcal{F} :

$!Site \vdash 1,$

$$\Gamma_{\mathcal{F}} = \begin{array}{l} \vdash SecureChannel \otimes Schedule \multimap_{bookFlight} Booking, \\ \vdash Password \multimap_{login} SecureChannel. \end{array}$$

Agent \mathcal{T} applies PD as follows:

$$\frac{Schedule \vdash Booking}{From \otimes To \vdash Booking} \mathcal{R}_f(findSchedule)$$

\mathcal{F} derives the proposal further:

$$\frac{\frac{Schedule \vdash Password \otimes Schedule}{Schedule \vdash SecureChannel \otimes Schedule} \mathcal{R}_b(login)}{Schedule \vdash Booking} \mathcal{R}_b(bookFlight)$$

Agent \mathcal{T} deduces the offer further:

$$\frac{Schedule \vdash Site \otimes Schedule}{Schedule \vdash Password \otimes Schedule} \mathcal{R}_b(\text{getPassword})$$

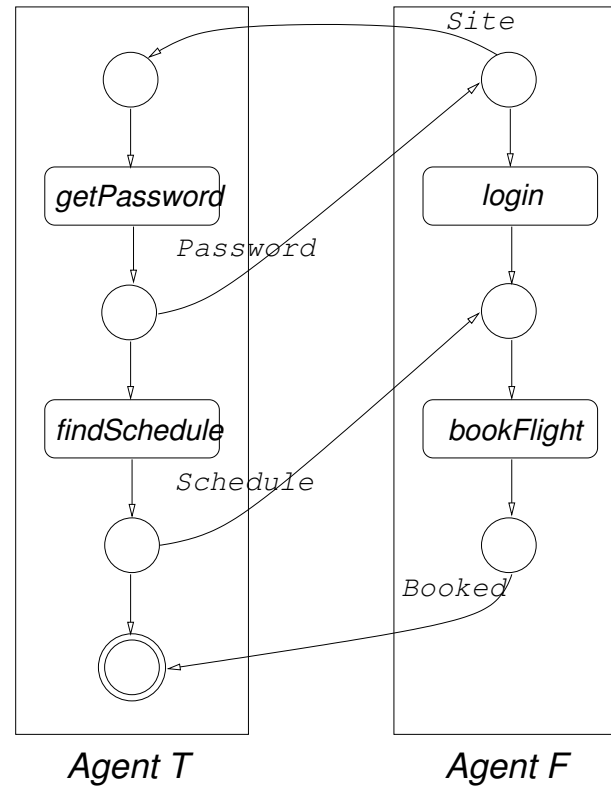
Now agent \mathcal{F} constructs a new offer based on its own specification:

$$\frac{Site \vdash 1}{!Site \vdash 1} \mathcal{R}_{L_i}$$

However, instead of forwarding it to \mathcal{T} , it merges the offer with the received complementary offer:

$$\frac{\frac{Site \otimes Schedule \vdash Site \otimes Schedule \quad Id \quad \overline{\vdash 1} \quad Axiom}{Site \otimes Schedule \vdash Site \otimes Schedule \otimes 1} \mathcal{R}_{\otimes}}{Site \otimes Schedule \vdash Site \otimes Schedule \otimes 1} \mathcal{R}_{\otimes}$$

An Example (Solution)



- A $\text{CSC} \vdash S' \multimap G'$ is executable, iff a $\text{CSC} \vdash S \multimap G$ is executable in a CSA $\Gamma; S \vdash G$ and there is a derivation $\vdash S \multimap G \Rightarrow_{\mathcal{R}} \dots \Rightarrow_{\mathcal{R}} \vdash S' \multimap G'$
- Since all our PD steps are described with LL inference figures, our PD formalism is sound

- A $\text{CSC} \vdash S \multimap G$ is executable, iff a $\text{CSC} \vdash S' \multimap G'$ is executable in a CSA $\Gamma; S' \vdash G'$ and there is a derivation $\vdash S \multimap G \Rightarrow_{\mathcal{R}} \dots \Rightarrow_{\mathcal{R}} \vdash S' \multimap G'$
- Since we generate all possible resultants, our formalism is complete

PD Strategies (Selection Criteria)

- Mixed backward and forward chaining
- Different search methods
- Prefer resultants with smaller derivation length
- Apply only one PD step at time
- Combine several PD steps together
- Priority-based selection—some literals are preferred

PD Strategies (Stopping Criteria)

- The derived resultant is computationally equivalent to a previous one
- A generative cycle is detected
- Maximum derivation length is reached
- The resultant is equal to the goal
- Stepwise
- Exhaustive—derivation stops, when no new resultants are available

- A broadcast-based architecture model in JADE/Java
- Downloadable at <http://www.idi.ntnu.no/~peep/symbolic>
- Used in teaching an agent course at NTNU

- We developed a model for symbolic negotiation with PD
- We proved soundness and completeness of PD for LL
- PD strategies have been proposed
- The propositional fragment of the proposed framework has been implemented

- Further experiments with PD strategies and protocols
- P2P implementation of symbolic negotiation
- Applications (i.e. composition of Semantic Web services)
- Combining symbolic negotiation and non-symbolic negotiation
- Operational semantics for the framework (binds together PD, PD strategies and protocols)

Thank you!