Computing Presuppositions by Contextual Reasoning

Christof Monz
Institute for Logic, Language and Computation (ILLC)
University of Amsterdam
Plantage Muidergracht 24, 1018 TV Amsterdam
The Netherlands
E-mail: christof@wins.uva.nl

Abstract

This paper describes how automated deduction methods for natural language processing can be applied more efficiently by encoding context in a more elaborate way. Our work is based on formal approaches to context, and we provide a tableau calculus for contextual reasoning. This is explained by considering an example from the problem area of presupposition projection.

Introduction

The notion of context plays an important role in formal theories of natural language processing (NLP), one of the major subareas of Artificial Intelligence. Several definitions of context have been used in computational linguistics, depending on the application. Amongst others, context has been used for resolving lexical ambiguity (Buvač 1996), pronoun resolution (Monz & de Rijke 1999), and presupposition projection (Karttunen 1974). In this paper, we focus on the last application.

Before we go into detail, we give a rough definition of presupposition. We say that an utterance φ presupposes a fact π, if uttering φ only makes sense if the context (e.g., world knowledge or earlier utterances in the same conversation) provides enough information to conclude that π is the case. Consider an example.

(1)a. Sue's husband is out of town.

b. If Sue is married, then her husband is out of town.

Presupposition triggers like *Sue's husband* are resolved against their local context. If this context provides the presupposed information, then the presupposition does not project. If, on the other hand, the local context does not provide the presupposed information, it does project.

As to the explicit representation of context itself, we try to keep things as simple as possible and identify a context with a set of first-order formulas.

There are many formal theories in Linguistics that describe presupposition projection by using context, e.g. (Karttunen 1974), but only a few NLP systems computing presuppositions have been actually implemented, see, e.g., DORIS (Blackburn *et al.* 1999). To compute the presuppositions which project, it is necessary to employ automated reasoning techniques, where, as we said before, a presupposition π does not project if it can be deduced from its local context. Computing presupposition projection involves both concepts: *context* and *reasoning*.

If a semantic representation of a natural language discourse contains the presuppositions $\pi_1, \ldots, \pi_n$, where $\Gamma_1, \ldots, \Gamma_n$ are the corresponding local contexts (sets of first-order formulas), then the most straightforward way to compute the projected presuppositions is by having run a theorem prover on the queries (or projection problems) $\Gamma_1 \vdash \pi_1, \ldots, \Gamma_n \vdash \pi_n$. Its major shortcoming is that deductions are carried out independently of each other. But, often it is the case that the contexts $\Gamma_1, \ldots, \Gamma_n$ share formulas. Then, some deduction steps are carried out several times, causing a decrease of efficiency.

To avoid redundant treatment of subcontexts, we need a richer language that enables us to express nesting of contexts. Here, we use the in-predicate, cf. (Attardi & Simi 1994a; 1994b), which takes two arguments. The first argument is a set of formulas and the second is a formula. $\mathtt{in}(\Gamma, \varphi)$ is true if $\Gamma \vdash \varphi$.[1] Since φ itself can contain an in-predicate, we are able to nest contexts. E.g., $\mathtt{in}(\Gamma, \varphi \wedge \mathtt{in}(\Delta, \psi))$ is true if $\Gamma \vdash \varphi$ and $\Gamma \cup \Delta \vdash \psi$. In this case Δ is a local context for ψ.

A language containing the in-predicate functions like a meta-language of reasoning. In the sequel, we give a tableau calculus for such a language.

Presupposition and Context

A particular instance of the presupposition projection problem was given in (1). Here, the noun phrase *Sue's husband* carries the presupposition *Sue is married.* In general, following (Blamey 1986), we use an additional connective π/φ, meaning that φ presupposes π.

Definition 1 (The Language $\mathcal{L}^{pre}$) To indicate the presuppositions of a formula, the language of first-order

[1] We slightly diverge from (Attardi & Simi 1994a; 1994b) where the argument positions of in are interchanged, i.e., $\mathtt{in}(\varphi, \Gamma)$ means that φ can be deduced from Γ.

predicate logic $\mathcal{L}$ is augmented with a binary presupposition connective '/'. Its left argument is a first-order formula and its right argument a formula of $\mathcal{L}^{pre}$.

$$\varphi ::= R(t_1 \ldots t_n) \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \rightarrow \psi \mid \varphi \vee \psi \mid \forall x\, \varphi \mid \exists x\, \varphi \mid \pi/\varphi$$

where $\pi \in \mathcal{L}$.

The set of contexts $\mathcal{C}$ is identified with the finite subsets of $\mathcal{L}$.

Above, we sketched how the projected presupposition of (1) can be computed. (Karttunen 1974) defines a function pres accomplishing this task.

Definition 2 (Presupposition) The function pres is defined recursively. It takes two arguments: A formula of $\mathcal{L}^{pre}$ and a context $\Gamma \in \mathcal{C}$, $\mathsf{pres} : \mathcal{L}^{pre} \times \mathcal{C} \rightarrow POW(\mathcal{L})$.

(i)	$\mathsf{pres}(\varphi)^\Gamma$	=	$\emptyset$ if φ is atomic
(ii)	$\mathsf{pres}(\neg\varphi)^\Gamma$	=	$\mathsf{pres}(\varphi)^\Gamma$
(iii)	$\mathsf{pres}(\varphi \wedge \psi)^\Gamma$	=	$\mathsf{pres}(\varphi)^\Gamma \cup \mathsf{pres}(\psi)^{\Gamma \cup \{\mathsf{as}(\varphi)\}}$
(iv)	$\mathsf{pres}(\varphi \rightarrow \psi)^\Gamma$	=	$\mathsf{pres}(\varphi)^\Gamma \cup \mathsf{pres}(\psi)^{\Gamma \cup \{\mathsf{as}(\varphi)\}}$
(v)	$\mathsf{pres}(\varphi \vee \psi)^\Gamma$	=	$\mathsf{pres}(\varphi)^\Gamma \cup \mathsf{pres}(\psi)^{\Gamma \cup \{\mathsf{as}(\neg\varphi)\}}$
(vi)	$\mathsf{pres}(\forall x\, \varphi)^\Gamma$	=	$\mathsf{pres}(\varphi)^\Gamma$
(vii)	$\mathsf{pres}(\exists x\, \varphi)^\Gamma$	=	$\mathsf{pres}(\varphi)^\Gamma$
(viii)	$\mathsf{pres}(\pi/\varphi)^\Gamma$	=	$\begin{cases} \mathsf{pres}(\varphi)^\Gamma \cup \{\pi\} & \text{if } \Gamma \nvdash \pi \\ \mathsf{pres}(\varphi)^\Gamma & \text{if } \Gamma \vdash \pi \end{cases}$

The last rule is the one we want to focus on this paper, because it involves contextual reasoning.

The function as returns the assertive content of a formula of $\mathcal{L}^{pre}$, cf. (Karttunen & Peters 1979). Roughly speaking, the assertive part of a formula $\varphi \in \mathcal{L}^{pre}$ is obtained by substituting all subformulas of the form π/ψ by ψ. The assertive part can be recursively computed by applying the function as to φ.

Definition 3 (Assertive Content) as is a function from $\mathcal{L}^{inc}$ to $\mathcal{L}$:

$$\begin{aligned} \mathsf{as}(R(t_1 \ldots t_n)) &= R(t_1 \ldots t_n) \\ \mathsf{as}(\neg\varphi) &= \neg\mathsf{as}(\varphi) \\ \mathsf{as}(\varphi \bullet \psi) &= \mathsf{as}(\varphi) \bullet \mathsf{as}(\psi), \text{ where } \bullet \in \{\wedge, \rightarrow, \vee\} \\ \mathsf{as}(\pi/\varphi) &= \mathsf{as}(\varphi) \\ \mathsf{as}(\mathcal{Q}x\, \varphi) &= \mathcal{Q}x\, \mathsf{as}(\varphi), \text{ where } \mathcal{Q} \in \{\forall, \exists\} \end{aligned}$$

Let us consider an example. (2) displays an admittedly quite artificial discourse, but we hope that it illustrates the point.

(2)a. John is married. (φ)

b. If John's wife has a brother, then John hasn't met his brother-in-law, yet. ($\varphi/\psi \rightarrow (\varphi \wedge \psi)/\chi$)

c. John's wife has quite a few relatives. (φ/ω)

The semantic representations of the three sentences are displayed in parentheses after each sentence and the presupposition triggers are underlined.

Applying pres to the whole discourse (the conjunction of (2.a), (2.b), and (2.c)), it correctly computes that no presupposition projects:

$$\mathsf{pres}(\varphi \wedge ((\varphi/\psi \rightarrow (\varphi \wedge \psi)/\chi) \wedge \varphi/\omega))^\emptyset = \emptyset$$

To see why this is the case, let us have a closer look at the way pres recursively computes the presuppositions. The contextual parameter of pres is initialized to $\emptyset$, as there is no further context preceding (2).

$\mathsf{pres}(\varphi \wedge ((\varphi/\psi \rightarrow (\varphi \wedge \psi)/\chi) \wedge \varphi/\omega))^\emptyset$

$= \mathsf{pres}(\varphi)^\emptyset \cup \mathsf{pres}((\varphi/\psi \rightarrow (\varphi \wedge \psi)/\chi) \wedge \varphi/\omega)^{\{\varphi\}}$ by (iii)

$= \mathsf{pres}((\varphi/\psi \rightarrow (\varphi \wedge \psi)/\chi) \wedge \varphi/\omega)^{\{\varphi\}}$ by (i)

$= \mathsf{pres}((\varphi/\psi \rightarrow (\varphi \wedge \psi)/\chi))^{\{\varphi\}} \cup \mathsf{pres}(\varphi/\omega)^{\{\varphi,\psi\}}$ by (iii)

$= \mathsf{pres}(\varphi/\psi)^{\{\varphi\}} \cup \mathsf{pres}((\varphi \wedge \psi)/\chi))^{\{\varphi,\psi\}} \cup \mathsf{pres}(\varphi/\omega)^{\{\varphi,\psi\rightarrow\chi\}}$ by (iv)

$= \mathsf{pres}((\varphi \wedge \psi)/\chi))^{\{\varphi,\psi\}} \cup \mathsf{pres}(\varphi/\omega)^{\{\varphi,\psi\rightarrow\chi\}}$ by (viii)
where $\mathsf{pres}(\varphi/\psi)^{\{\varphi\}} = \emptyset$ because $\{\varphi\} \vdash \varphi$

$= \mathsf{pres}(\varphi/\omega)^{\{\varphi,\psi\rightarrow\chi\}}$ by (viii)
where $\mathsf{pres}((\varphi \wedge \psi)/\chi)^{\{\varphi,\psi\}} = \emptyset$ because $\{\varphi,\psi\} \vdash \varphi \wedge \psi$

$= \emptyset$ by (viii)
where $\mathsf{pres}(\varphi/\omega)^{\{\varphi,\psi\rightarrow\chi\}} = \emptyset$ because $\{\varphi,\psi \rightarrow \chi\} \vdash \varphi$

The application of pres to (2) involves three inferences: $\{\varphi\} \vdash \varphi$, $\{\varphi,\psi\} \vdash \varphi \wedge \psi$, and $\{\varphi,\psi \rightarrow \chi\} \vdash \varphi$. Here, the premise φ is used three times and the proving method has to apply the same set of rules three times to the same formula. As we mentioned earlier, this is due to the fact that the contexts (premises) are considered to be independent of each other. From a computational point of view, this redundancy is rather inefficient. Of course, the example is very simple, but in general, φ could have been arbitrarily complex. In (2), the sentences containing presuppositions, (2.b) and (2.c), occur in a context consisting of one sentence, viz. (2.a), but it could have been a much larger context.

This reasoning task can be done more efficiently if we take the flow of contextual information into account; i.e., the way contexts are nested. To express nesting of contexts we use a language containing the in-predicate:

Definition 4 (The Language $\mathcal{L}^{con}$) $\mathcal{L}^{con}$ is defined recursively as follows, where $\Gamma \in \mathcal{C}$:

$$\varphi ::= R(t_1 \ldots t_n) \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \rightarrow \psi \mid \varphi \vee \psi \mid \forall x\, \varphi \mid \exists x\, \varphi \mid \mathtt{in}(\Gamma, \varphi) \mid \top$$

We do not use $\mathcal{L}^{con}$ to express the semantics of a natural language discourse, but only for expressing which presupposition triggers have to be evaluated against which context. The purpose of $\mathcal{L}^{con}$ is to express these projection problems in a non-redundant fashion.

Let us reconsider example (2). The antecedent of (2.b) contains the presupposition, that John is married, formalized as φ. The context of this presupposition only consists of (2.a), also formalized as φ. This projection problem can be stated in $\mathcal{L}^{con}$ as $\mathtt{in}(\{\varphi\}, \varphi)$. The presupposition of the succedent of (2.b), i.e., John is married and that his wife has a brother, formalized as $\varphi \wedge \psi$ is evaluated against (2.a) and the antecedent of (2.b): $\mathtt{in}(\{\varphi,\psi\}, \varphi \wedge \psi)$. The two projection problems can be compactly expressed by a single formula of $\mathcal{L}^{con}$: $\mathtt{in}(\{\varphi\}, \varphi \wedge \mathtt{in}(\{\psi\}, \varphi \wedge \psi))$. The second in-predicate is nested under a context where φ already holds, and it is not necessary to express again that φ holds. The

last sentence, (2.c), again presupposes φ. Its context is (2.a) and(2.b), therefore, $\mathtt{in}(\{\varphi, \psi \rightarrow \chi\}, \varphi)$ has to hold. Putting the three projection problems together, we get

(3) $\mathtt{in}(\{\varphi\}, \varphi \wedge \mathtt{in}(\{\psi\}, \varphi \wedge \psi) \wedge \mathtt{in}(\{\psi \rightarrow \chi\}, \varphi))$

Note, that we did not embed the third in-predicate in the second in-predicate, yielding

(4) $\mathtt{in}(\{\varphi\}, \varphi \wedge \mathtt{in}(\{\psi\}, \varphi \wedge \psi \wedge \mathtt{in}(\{\psi \rightarrow \chi\}, \varphi)))$

This is not possible because ψ is a local assumption (context) only accessible within the conditional.

The obvious question is how we get from a semantic representation in $\mathcal{L}^{pre}$ to a description of the projection problems in $\mathcal{L}^{con}$. To this end, we define a translation function τ which takes two arguments: a formula of $\mathcal{L}^{pre}$ and a context. Before we can give the formal definition of τ, we have to define how the potential presuppositions of a formula can be computed.

Definition 5 (Potential Presupposition) The potential presupposition of a formula $\varphi \in \mathcal{L}^{inc}$ are all formulas $\pi \in \mathcal{L}$ that occur as subformulas of φ of the form π/ψ.

$$\begin{aligned}
\mathsf{pp}(R(t_1 \ldots t_n)) &= \emptyset \\
\mathsf{pp}(\neg\varphi) &= \mathsf{pp}(\varphi) \\
\mathsf{pp}(\varphi \bullet \psi) &= \mathsf{pp}(\varphi) \cup \mathsf{pp}(\psi) \text{ if } \bullet \in \{\wedge, \rightarrow, \vee\} \\
\mathsf{pp}(\pi/\varphi) &= \mathsf{pp}(\varphi) \cup \{\pi\} \\
\mathsf{pp}(\mathcal{Q}x\,\varphi) &= \mathsf{pp}(\varphi) \text{ if } \mathcal{Q} \in \{\forall, \exists\}
\end{aligned}$$

pp simply collects all presuppositions without checking whether they are entailed by there local context.

Now, we can define a translation τ from $\mathcal{L}^{pre}$ to $\mathcal{L}^{con}$. The function τ is defined in Table 1. Since it is rather complex and we have only limited space, we just try to sketch its rationale. τ is defined recursively, taking two parameters: a formula of $\mathcal{L}^{pre}$ and a context Γ.

Let χ be in $\mathcal{L}^{pre}$. If the main operator is unary and χ contains potential presuppositions, then we proceed with translating the immediate subformula of χ. If χ is of the form $\varphi \wedge \psi$, we check whether φ contains potential presuppositions. If this is the case, χ translates as $\mathtt{in}(\Gamma, (\varphi)^{\tau,\emptyset} \wedge (\psi)^{\tau,\{\mathsf{as}(\varphi)\}})$. The first argument of in is the context and $(\varphi)^{\tau,\emptyset}$ gets $\emptyset$ as the context, because the translation of φ will be embedded in Γ by the in-predicate. Similarly for $(\psi)^{\tau,\{\mathsf{as}(\varphi)\}}$, but here the context is augmented by the assertive content of φ, which was not available for φ itself. No in-predicate is introduced if the context parameter is empty. Note that the way how contexts are augmented, see, for instance, $(\psi)^{\tau,\{\mathsf{as}(\varphi)\}}$, follows the definition of pres.

Reconsidering (2), the translation of its semantic representation in $\mathcal{L}^{pre}$

$$\varphi \wedge ((\varphi/\psi \rightarrow (\varphi \wedge \psi)/\chi) \wedge \varphi/\omega)$$

proceeds as follows:

$\tau : \mathcal{L}^{pre} \times \mathcal{C} \rightarrow \mathcal{L}^{con}$	
$R(t_1 \ldots t_n)^{\tau,\Gamma} = \top$	(1)
$(\neg\varphi)^{\tau,\Gamma} = \begin{cases} (\varphi)^{\tau,\Gamma} & \text{if } \mathsf{pp}(\varphi) \neq \emptyset \\ \top & \text{if } \mathsf{pp}(\varphi) = \emptyset \end{cases}$	(2a) (2b)
$(\varphi \bullet \psi)^{\tau,\Gamma}$ $= \begin{cases} \mathtt{in}(\Gamma, (\varphi)^{\tau,\emptyset} \wedge (\psi)^{\tau,\{\mathsf{as}(\varphi)\}}) & \text{if } \mathsf{pp}(\varphi) \neq \emptyset, \Gamma \neq \emptyset \\ (\varphi)^{\tau,\emptyset} \wedge (\psi)^{\tau,\{\mathsf{as}(\varphi)\}} & \text{if } \mathsf{pp}(\varphi) \neq \emptyset, \Gamma = \emptyset \\ (\psi)^{\tau,\Gamma \cup \{\varphi\}} & \text{if } \mathsf{pp}(\varphi) = \emptyset \end{cases}$ where $\bullet \in \{\wedge, \rightarrow\}$	(3a) (3b) (3c)
$(\varphi \vee \psi)^{\tau,\Gamma}$ $= \begin{cases} \mathtt{in}(\Gamma, (\varphi)^{\tau,\emptyset} \wedge (\psi)^{\tau,\{\mathsf{as}(\neg\varphi)\}}) & \text{if } \mathsf{pp}(\varphi) \neq \emptyset, \Gamma \neq \emptyset \\ (\varphi)^{\tau,\emptyset} \wedge (\psi)^{\tau,\{\mathsf{as}(\neg\varphi)\}} & \text{if } \mathsf{pp}(\varphi) \neq \emptyset, \Gamma = \emptyset \\ (\psi)^{\tau,\Gamma \cup \{\neg\varphi\}} & \text{if } \mathsf{pp}(\varphi) = \emptyset \end{cases}$	(4a) (4b) (4c)
$(\pi/\varphi)^{\tau,\Gamma} = \begin{cases} \mathtt{in}(\Gamma, \pi \wedge (\varphi)^{\tau,\emptyset}) & \text{if } \mathsf{pp}(\varphi) \neq \emptyset, \Gamma \neq \emptyset \\ \pi \wedge (\varphi)^{\tau,\emptyset} & \text{if } \mathsf{pp}(\varphi) \neq \emptyset, \Gamma = \emptyset \\ \mathtt{in}(\Gamma, \pi) & \text{if } \mathsf{pp}(\varphi) = \emptyset, \Gamma \neq \emptyset \\ \pi & \text{if } \mathsf{pp}(\varphi) = \emptyset, \Gamma = \emptyset \end{cases}$	(5a) (5b) (5c) (5d)
$(\mathcal{Q}x\,\varphi)^{\tau,\Gamma} = \begin{cases} \mathtt{in}(\Gamma, \mathcal{Q}x\,(\varphi)^{\tau,\emptyset}) & \text{if } \mathsf{pp}(\varphi) \neq \emptyset, \Gamma \neq \emptyset \\ \mathcal{Q}x\,(\varphi)^{\tau,\Gamma} & \text{if } \mathsf{pp}(\varphi) \neq \emptyset, \Gamma = \emptyset \\ \top & \text{if } \mathsf{pp}(\varphi) = \emptyset \end{cases}$ where $\mathcal{Q} \in \{\forall, \exists\}$	(6a) (6b) (6c)

Table 1: Translating from $\mathcal{L}^{pre}$ to $\mathcal{L}^{con}$

$(\varphi \wedge ((\varphi/\psi \rightarrow (\varphi \wedge \psi)/\chi) \wedge \varphi/\omega))^{\tau,\emptyset}$
by (3c) :
$= ((\varphi/\psi \rightarrow (\varphi \wedge \psi)/\chi) \wedge \varphi/\omega)^{\tau,\{\varphi\}}$
by (3a) :
$= \mathtt{in}(\{\varphi\}, (\varphi/\psi \rightarrow (\varphi \wedge \psi)/\chi)^{\tau,\emptyset} \wedge (\varphi/\omega)^{\tau,\{\psi \rightarrow \chi\}})$
by (3b) :
$= \mathtt{in}(\{\varphi\}, (\varphi/\psi)^{\tau,\emptyset} \wedge ((\varphi \wedge \psi)/\chi)^{\tau,\{\chi\}} \wedge (\varphi/\omega)^{\tau,\{\psi \rightarrow \chi\}})$
by (5d) :
$= \mathtt{in}(\{\varphi\}, \varphi \wedge ((\varphi \wedge \psi)/\chi)^{\tau,\{\chi\}} \wedge (\varphi/\omega)^{\tau,\{\psi \rightarrow \chi\}})$
by (5c) :
$= \mathtt{in}(\{\varphi\}, \varphi \wedge \mathtt{in}(\{\chi\}, \varphi \wedge \psi) \wedge (\varphi/\omega)^{\tau,\{\psi \rightarrow \chi\}})$
by (5c) :
$= \mathtt{in}(\{\varphi\}, \varphi \wedge \mathtt{in}(\{\chi\}, \varphi \wedge \psi) \wedge \mathtt{in}(\{\psi \rightarrow \chi\}, \varphi))$

Now we have defined an algorithmic way to state a presupposition projection problem in a compact way by translating it to $\mathcal{L}^{con}$. The main advantage of expressing projection problems in $\mathcal{L}^{con}$ is that it allows for a more efficient way of reasoning; see below.

Contextual Reasoning

In the previous section, we have seen how translating from $\mathcal{L}^{pro}$ to $\mathcal{L}^{con}$ can help stating presupposition projection problems in a compact way. In this section, we provide a tableau calculus for the language $\mathcal{L}^{con}$. If $\varphi \in \mathcal{L}^{con}$ is valid, then all presuppositions are entailed by their local context and none of them projects.

The most important rule of our tableau calculus $\mathcal{T}^{con}$ is the rule ($\neg$in). Before we introduce the other rules, it is helpful to have a closer look at ($\neg$in), in order to

understand the way context is represented in $\mathcal{T}^{con}$.

$$\frac{(i,\sigma):\neg\mathtt{in}(\{\varphi_1,\ldots,\varphi_n\},\psi)}{\begin{array}{c}(j,\sigma\cup\{i\}):\varphi_1\\ \vdots\\ (j,\sigma\cup\{i\}):\varphi_n\\ (j,\sigma\cup\{i\}):\neg\psi\end{array}}\ (\neg\mathtt{in})$$

To keep track of the contextual information, labels are attached to the nodes in the tableau. A label has two arguments. Its first argument i is a natural number ($i \in \mathbb{N}$), which is the identifier of the context. I.e., if two nodes have the same number as the first argument of their labels, then they belong to the same context.

The second argument σ is a set of natural numbers. This set contains the identifiers of the contexts that are accessible. We say that a context Γ is accessible from a formula ψ, if there is a formula of the form $\mathtt{in}(\Gamma,\varphi)$ and ψ is a subformula of φ. For instance, considering the formula $\mathtt{in}(\Gamma,\varphi \wedge \mathtt{in}(\Delta,\psi))$, Γ is accessible from φ and $\mathtt{in}(\Delta,\psi)$. Also Δ is accessible from ψ. Since accessibility is transitive, it holds that Γ is accessible from ψ; but Δ is not accessible from φ.

The $(\neg\mathtt{in})$-rule is similar to the upwards direction (entering a context) of the (CS)-rule in (Buvač & Mason 1993):

$$\frac{\vdash_{\bar{\kappa}*\kappa_1}\varphi}{\vdash_{\bar{\kappa}}\mathtt{ist}(\kappa_1,\varphi)}\ (\mathrm{CS})$$

$\bar{\kappa}$ represents a sequence of contexts and the upwards direction of the rule says that if it is true in the context $\bar{\kappa}$ that φ holds in the extension with κ_1, then φ holds in the context $\bar{\kappa} * \kappa_1$ itself.

Comparing (CS) to $(\neg\mathtt{in})$, we can say that $\bar{\kappa}$ corresponds to $\sigma \cup \{i\}$ and j, the identifier of the context extension with $\{\varphi_1,\ldots,\varphi_n,\neg\psi\}$, corresponds to κ_1.

Table 2 gives the complete set of tableau rules. Besides the rules for the in-predicate, it contains the usual rule for the boolean connectives and quantifiers. ℓ is a meta-variable over labels of the form (i,σ). As the expansion rules for the regular logical connectives do not change the contextual information, the label of the parent and the label(s) of the daughter(s) are identified.

The contextual information carried by the labels becomes important when we want to define the closure conditions of a branch.

Definition 6 (Closure of a Branch) A branch of a tableau tree is closed if it contains two nodes of the form $(i,\sigma):\varphi$ and $(j,\sigma'):\neg\psi$ such that

(a) φ and ψ are unifiable, and
(b) (i) $i = j$ or (ii) $i \in \sigma'$ or (iii) $j \in \sigma$

(a) is the standard condition on branch closure. (b) considers three cases. If $i = j$, then both literals belong to the same context. If $i \in \sigma'$, then φ belongs to an extension of j. (iii) is analogous to the previous one.

Again, we consider the projection problem of (2). Applying the tableau expansion rules to the negation of (3), we try to derive a contradiction (a closed tableau).

$$\frac{(i,\sigma):\mathtt{in}(\{\varphi_1,\ldots,\varphi_n\},\psi)}{(j,\sigma\cup\{i\}):\bigvee_{i=1}^{n}\neg\varphi_i \mid (j,\sigma\cup\{i\}):\psi}\ (\mathtt{in})$$

$$\frac{(i,\sigma):\neg\mathtt{in}(\{\varphi_1,\ldots,\varphi_n\},\psi)}{\begin{array}{c}(j,\sigma\cup\{i\}):\varphi_1\\ \vdots\\ (j,\sigma\cup\{i\}):\varphi_n\\ (j,\sigma\cup\{i\}):\neg\psi\end{array}}\ (\neg\mathtt{in})$$

$$\frac{\ell:\varphi_1\wedge\varphi_2}{\begin{array}{c}\ell:\varphi_1\\ \ell:\varphi_2\end{array}}\ (\wedge) \qquad \frac{\ell:\neg(\varphi_1\wedge\varphi_2)}{\ell:\neg\varphi_1\vee\neg\varphi_2}\ (\neg\wedge)$$

$$\frac{\ell:\varphi_1\vee\varphi_2}{\ell:\varphi_1 \mid \ell:\varphi_2}\ (\vee) \qquad \frac{\ell:\neg(\varphi_1\vee\varphi_2)}{\ell:\neg\varphi_1\wedge\neg\varphi_2}\ (\neg\vee)$$

$$\frac{\ell:\varphi_1\rightarrow\varphi_2}{\ell:\neg\varphi_1 \mid \ell:\varphi_2}\ (\rightarrow) \qquad \frac{\ell:\neg(\varphi_1\rightarrow\varphi_2)}{\ell:\varphi_1\wedge\neg\varphi_2}\ (\neg\rightarrow)$$

$$\frac{\ell:\neg\neg\varphi}{\ell:\varphi}\ (\neg\neg)$$

$$\frac{\ell:\forall x\,\varphi}{\ell:\varphi[x/X]}\ (\forall) \qquad \frac{\ell:\exists x\,\varphi}{\ell:\varphi[x/f(X_1\ldots X_n)]}\ (\exists)$$

$$\frac{\ell:\neg\forall x\,\varphi}{\ell:\exists x\,\neg\varphi}\ (\neg\forall) \qquad \frac{\ell:\neg\exists x\,\varphi}{\ell:\forall x\,\neg\varphi}\ (\neg\exists)$$

where for $(\mathtt{in})$ and $(\neg\mathtt{in})$: j is a fresh natural number that does not occur elsewhere in the tableau, $(\forall)$: X is free in φ, and $(\exists)$: $X_1\ldots X_n$ are the free variables in φ.

Table 2: The set of tableaux rules

To verify the negation of the projection problem in the initial context 0, we create a new context 1, where the contextual assumptions of the in-predicate hold, but not the second argument. Both nodes, the node belonging to the contextual assumption of the in-predicate and the negation of its second argument, carry a 1 as its context identifier and have 0 (the context in which the in-predicate occurred) as an element of the set of accessible contexts. The remaining steps either follow the same pattern or are just regular boolean tableau expansion rules. The pairs of nodes that allow to close a branch are connected by a dashed line. $(1,\{0\}):\varphi$ and $(2,\{1,0\}):\neg\varphi$ allow to close the second branch from the left, because the first node is accessible from the second, as $1 \in \{1,0\}$, which is the set of contexts that are accessible from the second node. The first and the third branch can be closed because both nodes belong to the same context. The two right-most branches can be closed, because 1 is accessible from 3.

An Application to Dialogue Systems

Dialogue systems such as route planning systems, e.g., Trindi (Cooper et al. 1999), require that the system

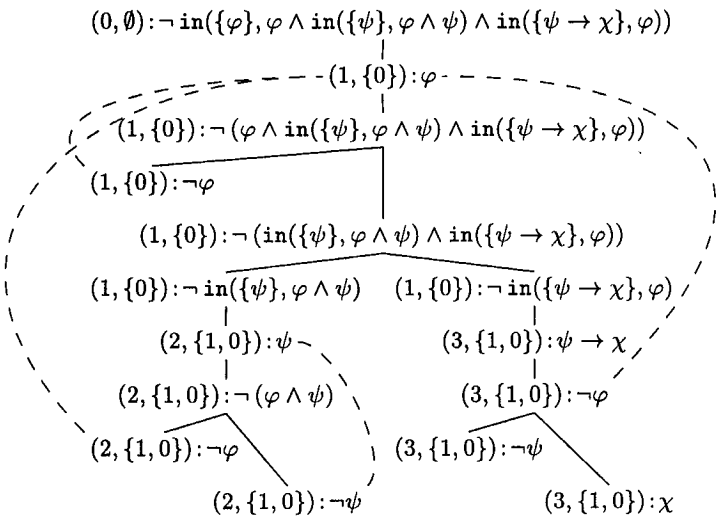

Figure 1: The tableau for the projection problem of (2)

can deal with presuppositions. In dialogues, presuppositions play an important role because it is convenient to take some things for granted as they have been mentioned before or they are part of common knowledge. For instance, in the following dialogue, the presupposition trigger *my start*, assumes that it a start-location has been mentioned before.

User: I would like to go to Paris.

System: Where do you start?

User: My start is Amsterdam

System: When do you travel? ...

In order to detect whether the user presupposes material that has not been mentioned in the preceding discourse and which is not part of the system's knowledge base, it is necessary that the system can compute those presuppositions that project. If a presupposition projects, the system can get back to the user and ask him for futher explanations. This way of detecting missing information by using deductive methods is more elegant than simply looking whether a term has occurred before. The system behaves more intelligent and has to ask less questions about things which are obvious for the user as he has told the system before—although in other words. In order to improve the acceptability of dialogue system it is mandatory that the user can convey information in a convenient and non-redundant way without having to repeat things.

Although the way context is represented in the Trindi system differs from our approach, it is possible to adapt our techniques to the Trindi system, cf. (Monz 1999).

Conclusions and Future Work

Computing the presuppositions of a natural language discourse is an important task for a natural language processing system. Employing a language like $\mathcal{L}^{con}$ allows for a non-redundant way of stating presupposition problems. To this end, we gave a translation from $\mathcal{L}^{pre}$ to $\mathcal{L}^{con}$. In addition, a tableau calculus $\mathcal{T}^{con}$ has been presented, which allows to compute presupposition projection more efficiently than approaches considering the projection problems independent of each other.

In this paper, contexts were simply identified with sets of first-order formulas. Our future work will focus on a more complex representation of context, as it recently emerged in computational linguistics, cf. (van der Sandt 1992). His approach seems to be more appropriate for describing some projection phenomena that cannot be explained within a framework that uses a simple notion of context like we did. Nevertheless, we think that the translation function and the tableau calculus are defined generally enough to be adaptable to a more refined representation of context.

Acknowledgments. Thanks go to Marco Aiello and Maarten de Rijke for helpful comments. The author was supported by the Physical Sciences Council with financial support from the Netherlands Organization for Scientific Research (NWO), project 612-13-001.

References

Attardi, G., and Simi, M. 1994a. Building proofs in context. In *Proc. of Meta '94*, LNCS 883, 410–424. Springer.

Attardi, G., and Simi, M. 1994b. Proofs in context. In Doyle, J., and Torasso, P., eds., *Principles of Knowledge Representation and Reasoning: Proc. of the Fourth International Conference*. Morgan Kaufmann.

Blackburn, P.; Bos, J.; Kohlhase, M.; and de Nivelle, H. 1999. Inference and computational semantics. In Bunt, H., and Thijsse, E., eds., *3rd International Workshop on Computational Semantics (IWCS-3)*, 5–21. Tilburg University.

Blamey, S. 1986. Partial logic. In Gabbay, D., and Guenthner, F., eds., *Handbook of Philosophical Logic*, volume III. Kluwer Academic Publishers. 1–70.

Buvač, S., and Mason, I. 1993. Propositional logic of context. In Fikes, R., and Lehnert, W., eds., *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 412–419. Menlo Park, California: AAAI Press.

Buvač, S. 1996. Resolving lexical ambiguity using a formal theory of context. In Peters, S., and van Deemter, K., eds., *Semantic Ambiguity and Underspecification*. CSLI Lecture Notes. 101–124.

Cooper et al., R. 1999. Coding instructional dialogue for information states. Deliverable D1.3, Trindi Project.

Karttunen, L., and Peters, S. 1979. Conventional implicatures in Montague Grammar. In Oh, C., and Dineen, D., eds., *Syntax and Semantics 11: Presupposition*. Academic Press, New York. 1–56.

Karttunen, L. 1974. Presupposition and linguistic context. *Theoretical Linguistics* 1(1):181–194.

Monz, C., and de Rijke, M. 1999. A tableau calculus for pronoun resolution. In Murray, N., ed., *Proc. of* TABLEAUX'99, LNAI. Springer.

Monz, C. 1999. Contextual inference in computational semantics. Submitted for publication.

van der Sandt, R. 1992. Presupposition projection as anaphora resolution. *Journal of Semantics* 9:333–377.