Continuations as a semantics-pragmatics interface for presuppositions

Timothée Bernard
Laboratoire de linguistique formelle / Université Paris Diderot-Paris 7
Sémagramme / Inria Nancy - Grand Est
timothee.bernard@ens-lyon.org

Introduction The projection problem for presupposition [Langendoen and Savin 1971] consists in determining the presupposition of complex expressions based on their parts. For instance, a presupposition of the consequent of a conditional may project as in (1a), be cancelled as in (1b), or weakened as in (1c) (which presupposes *If John is 64 years old, he can't be hired*).

(1) a. If the problem was difficult, then Morton isn't the one who solved it. [Soames 1982]
b. If the king has a son, the king's son is bald. [Heim 1983]
c. If John is 64 years old, he knows that he can't be hired. [Schlenker 2011.a]

Satisfaction theories, such as [Heim 1983], systematically predict that conditionals give rise to weak conditional presuppositions, for presuppositions triggered in the consequent and not entailed by the antecedent. This intuitively incorrect result (see (1a)) is what [Geurts 1996] dubbed the *Proviso Problem*. For this reason, pragmatic mechanisms that strengthen the predicted conditional presuppositions have been proposed (see for instance [von Fintel 2008]). Besides the fact that the efficiency of such mechanisms is contested [Geurts 1996, Mandelkern 2016], this strategy decomposes presuppositions into a rigorously formalised semantic component and a less formal pragmatic component, with the semantics-pragmatics interface not being fully specified. Other theories, such as DRT [van der Sandt 1992] or more recently dissatisfaction theory [Mandelkern 2016], do not suffer from the Proviso Problem, but then fail to convincingly explain cases in which a genuine conditional presupposition is observed, as in (1c) or (2).

(2) If you accept this job, will you let your family know that you're going to be working for a thug? [Schlenker 2011.b]

The goal What we propose is not a theory of presupposition *per se*, where this is understood to entail an explanation of why presuppositions arise and of why connectives and quantifiers affect presuppositions in certain ways; rather, we aim at giving a formal, descriptively accurate account of presuppositions. We propose a very expressive framework based on continuation semantics [de Groote 2006, Barker and Shan 2014] in which it is not only possible to simulate previous theories (such as the predictions of satisfaction theory or DRT) but also to get new, arguably more accurate, results, by modifying the lexical entries (connectives, quantifiers, presupposition plugs, etc.). This framework has the advantage of neatly integrating the pragmatic processes with the compositional interpretation, as algorithms which, when specified, provide precise predictions that can be tested.

Continuations Continuations were first studied in the theory of programming languages and are related to the idea of order of computation. It is interesting to note that the type raising used in [Montague 1973], for instance, can be seen as a limited form of continuation technique [Barker 2004]. Using continuations, [de Groote 2006] showed how dynamic behaviour can be expressed compositionally without the need for dynamic semantics. This is possible because each term is able to read from, and update, a context variable. In our case, a context c consists of a list of formula $\phi_1, \cdots \phi_n$ and represents the formula $\exists x_1, \cdots x_m.\ \phi_1 \wedge \cdots \wedge \phi_m$ where the x_i are the free variables of the ϕ_j.

Exceptions Building on [de Groote and Lebedeva 2010, Lebedeva 2012], we implement presupposition projection and accommodation *via* an exception mechanism. An exception is

a special kind of object, common in programming languages, that when *raised*, interrupts the current computation and is transmitted to the closest scoping *handler*. The presupposition trigger *know* in (3) tries to prove its presupposition — "$Pc\,\mathbf{stop}$", the logical proposition resulting from evaluating its complement P in context c with the empty continuation $\mathbf{stop} = \lambda c.\ \top$ — from context c; when this succeeds, the assertive content "$know(s, Pc\,\mathbf{stop})$" is produced and the continuation ϕ is executed with an updated context; otherwise, an exception containing a reference to the presupposition is raised.

(3) $[\![\text{know}]\!] = \lambda PS.\ S(\lambda sc\phi.\ \texttt{if}(c \vDash (Pc\,\mathbf{stop})) \texttt{:}\ know(s, Pc\,\mathbf{stop}) \wedge \phi(know(s, Pc\,\mathbf{stop}) :: c)$
$\texttt{else: raise}\ \mathbf{Presupposition}(P))$

As mentioned above, an exception projects through the computation tree (i.e., the semantic tree) until it is caught by a handler. For instance, the following **gacc** term — an occurrence of which scopes over each sentence S — executes S and catches any projecting presupposition P to accommodate it, i.e., to compute P with S as part of its continuation. Note that because P is accommodated, its content will be added to the context, so that the presupposition trigger is trivially able to prove it during the second computation and will not raise the exception again.

(4) $\mathbf{gacc} = \lambda Sc\phi.\ (Sc\phi)\ \texttt{handle}\ \mathbf{Presupposition}(P)\ \texttt{with}\ \mathbf{gacc}\,Pc(\lambda c'.\ Sc'\phi)$

Conditionals Instead of the term $[\![\text{if}]\!]_1$ below from [Lebedeva 2012], we model conditionals with $[\![\text{if}]\!]_2$ (in the context update, we use "$[\cdots]$" as a shorthand for "$\neg(Ac(\lambda c'.\ \neg Bc'\mathbf{stop}))$").

(5) $[\![\text{if}]\!]_1 = \lambda ABc\phi.\ \neg(Ac(\lambda c'.\ \neg Bc'\mathbf{stop})) \wedge \phi([\cdots] :: c)$

(6) $[\![\text{if}]\!]_2 = \lambda ABc\phi.\ \neg(Ac(\lambda c'.\ (\neg Bc'\mathbf{stop})\ \texttt{handle}\ \mathbf{Presupposition}(P)\ \texttt{with if}\ choice_{if}(c,P)\texttt{:}$
$\texttt{raise}\ \mathbf{Presupposition}(P)\ \texttt{else: raise}\ \mathbf{Presupposition}([\![\text{if}]\!]_1 AP))) \wedge \phi([\cdots] :: c)$

This term contains a place-holder for an algorithm $choice_{if}$ whose task is to decide whether to let a presupposition of the consequent P project or to weaken it as a conditional presupposition. A trivial algorithm always answering *true* would lead to a systematically strong presupposition as in DRT; always answering *false* results in the Proviso Problem as in satisfaction theory. Importantly, a non-trivial (and cognitively plausible) $choice_{if}$ can use the context c, giving rise to the prediction that the presupposition in (1c) has to be weakened, but not in a context such that *Most applicants simply don't have the qualifications for the job, which they are usually too young to realise*; thus achieving a level of context-dependence similar (at least) to the one of [Lassiter 2012], while being able to clearly identify which presupposition is selected.

Discussion We have seen with conditionals how our framework, in which all components are fully specifiable, can simulate the predictions of previous theories. In a similar way, we are able to conveniently express how a quantifier (*a*, *each*, *none*) can make a quantified version of a presupposition project (we are not limited by DRT's trapping constraint). Also, because we are dealing with provability of logical formulas and not with sets of possible worlds, we are free from the usual issues related to logical truth and logical equivalence. For instance, if the theorem prover invoked with $\vDash$ in (3) above has limited capacity (a reasonable assumption), then one obtains that an uninformed hearer will interpret (7) as presupposing that there are infinitely many primes, even though this proposition is a logical truth (so always entailed by the context).

(7) Mary knows that there are infinitely many primes.

References Barker 2004, bit.ly/2oUuIxF; Barker and Shan 2014, bit.ly/2zcrN7E; von Fintel 2008 PP 22(1), doi.org/bgmzc5; Geurts 1996 L&P 19(3), doi.org/dmskhq; de Groote 2006, doi.org/cfvq; de Groote and Lebedeva 2010, bit.ly/2Fm4Vc6; Heim 1983, doi.org/

ckgs4n; Langendoen and Savin 1971, bit.ly/2trDCaN; Lassiter 2012 S&P 5, doi.org/hw3; Lebedeva 2012, bit.ly/2hC44WX; Mandelkern 2016, doi.org/ck8g; Montague 1973, doi.org/d8rqzc; van der Sandt 1992, JoS 9(4), doi.org/fsgmqk; Schlenker 2011.a L&LC 5(12), doi.org/cb8zms; Schlenker 2011.b 5(12), doi.org/dm44k2; Soames 1982 LI 13(3), bit.ly/2tx9vyy