A Calculus for Direct Deduction with Dominance Constraints

Jan Jaspars, University of Amsterdam
Alexander Koller, University of the Saarland

Abstract. Underspecification has become a popular approach to dealing with ambiguity. An important operation in this context is *direct deduction*, deduction on underspecified descriptions which is justified by the meaning of the described formulae. We instantiate an abstract approach to direct deduction to dominance constraints, a concrete underspecification formalism, and obtain a sound and complete calculus.

1 Introduction

Over the past few years, *underspecification* has become a popular approach to dealing with some types of ambiguities, especially in semantics. It tries to avoid or postpone the combinatorial explosion of readings in a sentence with multiple ambiguities by specifying the set of readings with a single, compact *description*. Semantic processing, such as deduction, should then take place on the descriptions instead of all the readings; readings are only enumerated by need.

An essential operation that must be possible on underspecified descriptions for underspecification to be attractive is *direct deduction*: given an underspecified description, derive a description of logical consequences of the described first-order formulas. For example, it should be possible to make the following simple derivation without resolving the scope ambiguity in the first premise:

Every man loves a woman.
John is a man.

John loves a woman.

There has been some debate over the correct notion of underspecified entailment that a calculus for direct deduction should model [Rey95, vD96, Pin99], without reaching a clear conclusion so far. As a way out of this, [Jas99] proposed a very modular calculus on generic underspecified descriptions that is based on a semantics which can be tailored to make several different proof rules sound and complete.

In this paper, we adapt the calculus from [Jas99] to the language of *dominance constraints* [KNT98], a tree logic which has been used for underspecified descriptions [ENRX98] and whose formal properties are well understood. This combination is interesting from both perspectives: From the perspective of [Jas99], it's the first instantiation of the generic approach to a "real" underspecification formalism, and from the perspective of [KNT98], it provides a direct deduction calculus, and it illustrates the power of the tree description approach.

After illustrating the basic idea of underspecification in Section 2, we start by presenting the abstract, general calculus in Section 3. This is complemented by a brief introduction to dominance constraints in Section 4. We present the concrete calculus for dominance constraints in Section 5 and conclude in Section 6.

2 Underspecification

A notorious problem in natural language processing is that the number of readings of an ambiguous sentence grows exponentially with the number of ambiguities, increasing the cost of later processing steps by an exponential factor. Semantic underspecification attempts to tackle the problem of combinatoric explosion of readings

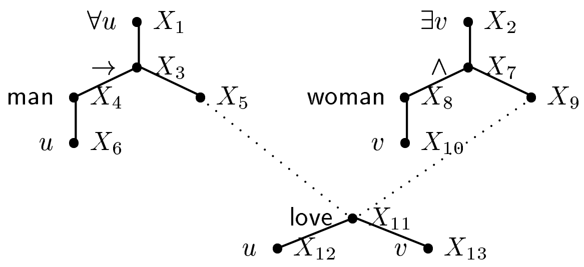

Figure 1: Underspecified description of the semantics of "Every man loves a woman".

by deriving from the syntax a single, compact *description* of the readings and working with descriptions for as long as possible. Direct deduction, then, is an operation on the descriptions that models deduction on the readings.

The new level of descriptions is usually called the *meta level* (ML), whereas the original level of (unambiguous) semantic representations is called the *object level* (OL). We will use predicate logic as the object language and dominance constraints as the meta language in this paper. To avoid confusion, we rename the OL connectives to $\underline{\wedge}$, $\underline{\forall}$, etc. Note that the abstract algorithm in Section 3 is independent of the exact choice of meta and object language.

An example for an underspecified representation can be found in Fig. 1. This graph describes the two readings of the sentence "Every man loves a woman", given in (1) and (2).

(1) $\forall u.\mathsf{man}(u) \to (\exists v.\mathsf{woman}(v) \wedge \mathsf{love}(u,v))$

(2) $\exists v.\mathsf{woman}(v) \wedge (\forall u.\mathsf{man}(u) \to \mathsf{love}(u,v))$

Informally, the description specifies the three "fragments" that must be put together to obtain a reading, plus constraints on the way they must be combined. The dotted lines can be thought of as "outscopes" relations; the same occurrence of the love subformula must be in the scope of both quantifiers. Because formulas have a tree-like structure in that each subformula is in the immediate scope of only one other formula, this means that the two quantifier fragments must take scope over each other. We will exploit this correspondence between formulas and trees in Section 4 to give the graph a formal meaning.

3 A Minimal Calculus of Direct Deduction

In this section, we present an abstract, general system of direct deduction. We first introduce a very flexible notion of underspecified entailment and then present a representative sample of the proof rules. For details, see [Jas99].

Entailment. Direct deduction is an operation on underspecified descriptions which is justified by the meanings of the described formulas. This means that any system of direct deduction must integrate two different systems: the resolution of underspecified descriptions to the readings they describe, and an inference system for reasoning on the object level.

Semantically, this means that an entailment notion for direct deduction must combine the reading enumeration function `r` that assigns to a description the set of its readings and the usual OL interpretation function I, roughly as in the following picture:

$$
\begin{tikzcd}
ML \arrow[r, "\mathtt{r}"] \arrow[drr, dashed, "I^*"'] & \wp(OL) & \\
 & OL \arrow[r, "I"] & \wp(\text{models of } OL)
\end{tikzcd}
$$

As the foundation of our calculus, we define a very flexible notion of underspecified entailment which combines the OL models with *partial disambiguation functions* (pdfs). A pdf $\mathtt{d}$ is a function which maps a ML-formula φ to a subset of the full set of readings described by φ, that is, $\mathtt{d}\varphi \subseteq \mathtt{r}\varphi$; and $\mathtt{d}\varphi = \emptyset$ iff $\mathtt{r}\varphi = \emptyset$. Intuitively, a pdf $\mathtt{d}$ represents a moment of interpretation at which certain readings described by φ are excluded, namely $\mathtt{r}\varphi - \mathtt{d}\varphi$.

An integrating interpretation I^* of an ML-formula φ is then obtained by combining OL-models with pdfs in the following way:

$$
\begin{aligned}
&ML \xrightarrow{I^*} \wp(\text{models of } OL \times \text{pdfs}) \\
&I^*\varphi = \{\langle M, \mathtt{d}\rangle \mid M \in I\alpha \text{ for all } \alpha \in \mathtt{d}\varphi\}
\end{aligned}
$$

In other words, φ is *true* at $\langle M, \mathtt{d}\rangle$ whenever all the readings described by φ that are not excluded by $\mathtt{d}$ are true at M. Entailment is then established in the standard fashion: $\varphi \models \varphi'$ iff $I^*\varphi \subseteq I^*\varphi'$.[1]

Direct deduction. A sound and complete calculus for this entailment notion can be obtained by extending a standard structural Gentzen calculus for the OL. The calculus consists of the following rules:

1. All the structural rules from the underlying calculus: reflexivity, transitivity, and monotonicity.[2]
2. The connective rules from the underlying calculus, lifted to unambiguous descriptions.
3. Two *total disambiguation* rules.

As an example for a connective rule, consider the standard rule $\underline{\wedge}R$, which introduces conjunctions on the right hand side of a sequent. Its counterpart in our calculus looks as follows (we say that φ is a unique description of the OL formula A if A is the only formula described by φ).

$$
\begin{array}{ll}
\varphi_1, \varphi_2 \vdash \varphi \qquad & \varphi_1, \varphi_2 \text{ unique descriptions of } A_1, A_2, \\
 & \varphi \text{ unique description of } A_1 \underline{\wedge} A_2.
\end{array} \tag{3}
$$

The total disambiguation rules look as follows:

$$
\begin{array}{ll}
\varphi_1, \ldots, \varphi_n \vdash \varphi \qquad & \varphi_1, \ldots, \varphi_n \text{ unique descriptions} \\
\varphi \vdash \varphi_1, \ldots, \varphi_n & \text{of the elements of } \mathtt{r}\varphi
\end{array} \tag{4}
$$

These sequents represent minimal disambiguation rules of which the first says that if (descriptions of) all the readings can be derived, then the ML formula describing this set can be derived as well. The second sequent says that from such a ML formula, the disjunction of its readings can be derived.

All the rules taken together form a calculus which is sound and complete for the pdf-semantics of ML [Jas97, Jas99].

[1] Earlier notions of underspecified entailment, which didn't allow partial disambiguation functions, led to notions of entailment that are either stronger or weaker than what is intuitively plausible [vD96]. The definition here is similar in spirit to a proposal in [Rey95] in that this entailment relation synchronizes partial disambiguation of condition and conclusion.

[2] In the case of 'realistic' linguistic inference, where pragmatic preferences come on stage, one may expect the loss of structural rules such as reflexivity [Pin99] and monotonicity on the ML-level.

Ambiguous descriptions. These rules are not very practical or efficient for direct deduction with dominance constraints since each disambiguation step requires full resolution, i.e., full specification of all the readings, of a ML formula. The calculus and semantics above can be tailored to allow stronger inference rules that can be used on descriptions with more than one reading.

The stronger variants of the total disambiguation rules (4) are the following *partial disambiguation rules*:

$$\varphi_1, \ldots, \varphi_n \vdash \varphi \quad \text{and} \quad \varphi \vdash \varphi_1, \ldots, \varphi_n \qquad \text{if } \bigcup_{i=1}^{n} \mathtt{r}\varphi_i = \mathtt{r}\varphi \tag{5}$$

Note that the applicability of this rule talks about the reading sets of the formulas appearing in the sequent, but this does *not* necessarily mean that it is needed to enumerate the readings to check applicability. We will illustrate an alternative in Section 5.

The two rules are unsound for the class of arbitrary pdfs. Soundness and completeness can be regained by restricting the class of admissible pdfs. The first rule corresponds to so-called *left monotonic* pdfs, whereas the second corresponds to *right monotonic* pdfs. A pdf $\mathtt{d}$ is left monotonic if whenever $\cup_{i=1}^{n}\mathtt{r}\varphi_i = \mathtt{r}\varphi$, there is an i such that $\mathtt{d}\varphi_i \subseteq \mathtt{d}\varphi$. A pdf $\mathtt{d}$ is right monotonic if under the same condition it holds that $\mathtt{d}\varphi \subseteq \cup_{i=1}^{n}\mathtt{d}\varphi_i$.

Similarly, connective rules can be generalized to apply to ambiguous descriptions too. For example, the conjunction rule (3) may be replaced by

$$\varphi_1, \varphi_2 \vdash \varphi \qquad \mathtt{r}\varphi = \{A \underline{\land} B \mid A \in \mathtt{r}\varphi_1, B \in \mathtt{r}\varphi_2\} \tag{6}$$

The corresponding restriction for the pdfs $\mathtt{d}$ is that $\mathtt{r}\varphi = \{A \underline{\land} B \mid A \in \mathtt{r}\varphi_1, B \in \mathtt{r}\varphi_2\}$ must imply $\mathtt{d}\varphi \subseteq \{A \underline{\land} B \mid A \in \mathtt{d}\varphi_1, B \in \mathtt{d}\varphi_2\}$. Generalizations for the other connective rules can be obtained in an analogous manner [Jas99].

4 Dominance Constraints

Now we present *dominance constraints*, a concrete formalism which has recently been proposed as a tool for semantic underspecification [ENRX98, KNT98]. Dominance constraints are a tree logic: a logic whose models are trees. It can be applied to natural-language semantics by considering OL formulas as trees and taking a dominance constraint φ to describe the formulas corresponding to the trees that satisfy φ. Trees are ground terms over a signature Σ of function symbols $f, g, \underline{\land}, \underline{\forall}, \mathsf{man}, \ldots$ with fixed arities.

The syntax of dominance constraints uses *node variables* $X, Y, \ldots$ and is defined as follows:

$$\varphi ::= X{:}f(X_1, \ldots, X_n) \mid X \lhd^* Y \mid \varphi \land \varphi'.$$

Dominance constraints are interpreted over trees τ and variable assignments α into the set of nodes of τ. A *labeling constraint* $X{:}f(X_1, \ldots, X_n)$ is satisfied by such a pair iff the node denoted by X is labeled with f and its children are denoted by $X_1, \ldots, X_n$. An *(atomic) dominance constraint* $X \lhd^* Y$ is satisfied iff X is mapped to a node which (reflexively, transitively) dominates the node denoted by Y. Logical connectives are interpreted in the usual way. For underspecified descriptions, we only use the (purely conjunctive) *constraint language*, but here we will also use the full first-order language, whose validity problem is decidable [KNT98].

We write equality $X{=}Y$ as an abbreviation for $X \lhd^* Y \land Y \lhd^* X$, $\mathcal{V}(\varphi)$ for the set of variables, and $\bar{\exists}\varphi$ for the existential closure of a dominance constraint φ.

A *constructive solution* of a dominance constraint φ is a pair (τ, α) that satisfies φ and has the property that every node in τ is denoted by a variable X for which a labeling constraint $X{:}f(\dots)$ is in φ. Intuitively, constructive solutions don't contain "new material". In the first-order language, we can enforce constructivity of the models of a constraint φ by considering instead the models of the formula

$$\varphi^C \quad \equiv \quad \varphi \wedge \forall Z. \bigvee_{X:f(\dots) \text{ in } \varphi} X{=}Z.$$

Dominance constraints describe their constructive solutions. So if we consider Fig. 1 as a graphical notation for a dominance constraint by reading the nodes as variables, the dotted lines as dominance constraints and the solid lines and labels as labeling constraints, the diagram gets a formal meaning as a dominance constraints describing exactly the two readings.

Finally, we can express that a constraint φ has *exactly one* constructive solution by requiring that

1. $\neg\bar{\exists}\varphi^C$ is not valid (that is, φ^C has at least one solution);
2. $\exists Y_1' \dots \exists Y_n' \forall Y_1 \dots \forall Y_n.(\varphi^C \to \bigwedge_{Y \in \mathcal{V}(\varphi)} Y{=}Y')$ is valid, where $\mathcal{V}(\varphi) = \{Y_1, \dots, Y_n\}$, and $Y_1', \dots, Y_n'$ are new variables.

The second condition expresses that there is at most one constructive solution. It literally says that there is only one possible variable assignment that satisfies φ^C. In a constructive solution, variable assignments are surjective, so this fixes the domain of a satisfying tree. Finally, each label is enforced by constructivity as well.

5 Direct Deduction with Dominance Constraints

Now we instantiate the abstract algorithm from Section 3 to the concrete underspecification formalism from Section 4. That is, we take the language of dominance constraints as the meta language ML; if φ is a dominance constraint, then, $\mathbf{r}\varphi$ is the set of constructive solutions of φ.

The abstract calculus in Section 3 consists of three different types of rules: structural rules, connective rules, and disambiguation rules. The structural rules are independent of the exact ML and OL, so we don't need to say anything about them here. Below, we encode the other two kinds of rules using validity problems of first-order dominance formulas.

To check the applicability condition of the more flexible conjunction rules (6), we must test (1) whether all constructive models of a constraint φ have the conjunction symbol $\underline{\wedge}$ as their root labels and (2) whether two other constraints, φ_1 and φ_2, are descriptions of the conjuncts. This is the case iff there is a variable $R \in \mathcal{V}(\varphi)$ such that

1. $\mathcal{V}(\varphi) = \mathcal{V}(\varphi_1) \uplus \mathcal{V}(\varphi_2) \uplus \{R\}$;
2. $\models \bar{\forall}\exists R_1 \exists R_2 (\varphi^C \leftrightarrow ({\varphi_1}^C \wedge {\varphi_2}^C \wedge R{:}\underline{\wedge}(R_1, R_2) \wedge \bigwedge_{Y \in \mathcal{V}(\varphi_i), i=1,2} R_i \triangleleft^* Y))$.

We get the original connective proof rules (3), which only work on non-ambiguous descriptions, by further requiring that both φ_1 and φ_2 only have a single constructive solution.

Of the two types of disambiguation rules, we consider the partial disambiguation rules (5) first. In order to test applicability of this rule, we must check whether the set of constructive solutions of a constraint φ is the same as the union of the sets of constructive solutions of the constraints $\varphi_1, \ldots, \varphi_n$. This is the case iff the following validity holds:

$$\models \bar{\exists}\varphi^C \leftrightarrow \bigvee_i \bar{\exists}\varphi_i{}^C.$$

If we stick to the less powerful total disambiguation rules (4), we additionally require that each of the constraints $\varphi_1, \ldots, \varphi_n$ has a single constructive solution.

Validity of first-order dominance formulas is decidable, hence so are the applicability conditions of the proof rules. This means that we have defined a calculus for direct deduction with dominance constraints which is sound and complete with respect to the semantics defined in Section 3.

6 Conclusion

We have presented a sound and complete calculus for direct deduction with dominance constraints. It is an instantiation of the abstract calculus from [Jas99] and inherits its definition of underspecified entailment via partial disambiguations. The underspecification logic we have considered are tree descriptions with dominance constraints [ENRX98].

The proof rules require checking the validity of first-order formulas over dominance constraints, which is known to be decidable. But the complexity of these tests is non-elementary; furthermore, the proof rules can't be used to generate subformulas in proof search. So there is no straightforward way of building a theorem prover from this calculus.

This, of course, is an essential goal of future work. The most interesting aspect is probably to find versions of the partial disambiguation rules (5) that allow proof search. As it stands, all possible splittings of descriptions into more specific descriptions must be allowed to guarantee completeness. A more feasible version might obtain one specific partial disambiguation from a dominance constraint solver, building on previous work about disambiguation of dominance constraints, and still be complete.

References

[ENRX98] M. Egg, J. Niehren, P. Ruhrberg, and F. Xu. Constraints over Lambda-Structures in Semantic Underspecification. In *Proceedings COLING/ACL'98*, Montreal, 1998.

[Jas97] J. Jaspars. Minimal logics for reasoning with ambiguous expressions. CLAUS Report 94, Universität des Saarlandes, Saarbrücken, 1997.

[Jas99] J. Jaspars. Structural logics for reasoning with underspecified representations. In *Proceedings of ICOS-1*, ILLC, Amsterdam, 1999.

[KNT98] A. Koller, J. Niehren, and R. Treinen. Dominance constraints: Algorithms and complexity. In *Proceedings of LACL*, Grenoble, 1998.

[Pin99] M. Pinkal. On semantic underspecification. In *Proceedings of IWCS-3*, Tilburg, 1999.

[Rey95] U. Reyle. On reasoning with ambiguities. In *Proceedings of EACL*, Dublin, 1995.

[vD96] K. van Deemter. Towards a logic of ambiguous expressions. In Kees van Deemter and Stanley Peters, editors, *Semantic Ambiguity and Underspecification.* CSLI Publications, Stanford, 1996.