Abductive language interpretation as bottom-up deduction

Henning Christiansen

Roskilde University, Computer Science Dept.
P.O.Box 260, DK-4000 Roskilde, Denmark
E-mail: `henning@ruc.dk`

Abstract. A translation of abductive language interpretation problems into a deductive form is proposed and shown to be correct. No meta-level overhead is involved in the resulting formulas that can be evaluated by bottom-up deduction, e.g., by considering them as Constraint Handling Rules. The problem statement may involve background theories with integrity constraints, and minimal contexts are produced that can explain a discourse given.

Keywords: Abduction-as-deduction, language interpretation, context comprehension.

1 Introduction

We consider logical grammars that relate the production of phrases to a context. Grammar rules include conditions so that only phrases that are faithful to the context can be composed, where the context is represented by some theory. Consider as an example the following formula that is the logical form of the Definite Clause Grammar [22] rule `ab --> a, b, {`F`}`.

$$\mathtt{a}(x,y) \land \mathtt{b}(y,z) \land F \to \mathtt{ab}(x,z) \tag{1}$$

If two subphrases referred to by `a` and `b` have been recognized and the context condition F holds, it is concluded that an `ab` phrase is feasible, grammatically as well as with respect to the context. Language analysis with such rules works quite well when context is given in advance and a given discourse is checked to be syntactically and semantically sound. This paper concerns the extended problem referred to as *language interpretation* of finding proper context theory so that an analysis of an observed discourse is possible. In other words, the F in (1) refers to an unknown and initially empty component, and deductive reasoning with (1) as it is, is not of much use. Our approach involves a transformation of the rules by moving context references into the other side of the implication; for (1) as follows.

$$\mathtt{a}(x,y) \land \mathtt{b}(y,z) \to F \land \mathtt{ab}(x,z) \tag{2}$$

Obviously, (1) and (2) are not logically equivalent, but an intuitive reading of (2) as a productive rule gives a clue: If suitable `a` and `b` are found, it is feasible to assert F and (thus, under this assumption) to conclude `ab`.

A grammar as depicted by (1) can be though of as part of a *speaker's* capabilities, embedding his knowledge about the context into language, whereas (2) is relevant for a *listener* who wants to gain new context knowledge by an interpretation of the spoken.

Section 2 is the theoretical core of the paper formalizing this transformation and proving it correct with respect to a definition of language interpretation problems. In section 3 the model is extended with background theories that may include integrity constraints. No assumption is made about a particular grammar formalism.

Context theories can be found by a forward, bottom-up evaluation of the transformed grammar and background theory with the discourse introduced as as an axiom. The transformed formulas can be executed directly in the language of Constraint Handling Rules [17] (CHR) as explained in section 4, with CHR applied as grammar formalism using a notation introduced in [8]. An example of this is given in section 5. The paper ends with an account on related work and a few concluding remarks and suggestions for future work; section 6.3 makes the interesting conclusion that the core of our abduction method in CHR also can be applied together with top-down execution of DCGs.

An implementation of what we have described is available on the web [7].

2 An abductive model of language interpretation and its reformulation as deduction

First-order logic is assumed and for given set of formulas Γ, closure(Γ) refers to a maximal set of ground atoms that follow deductively from Γ.

The vocabulary for a language interpretation problem consists of disjoint sets of predicates referred to as *grammar symbols* and *context predicates*. Grammar symbols are separated into *token level* symbols and *phrase level* symbols.

No detailed assumptions are made for the grammars applied, they can in principle apply any kind of transformations, multiple passes and be based on trees, graphs or something completely different. The basic components in a language interpretation scenario are the following.

Discourse: A set of atomic and ground token level atoms giving the set of input tokens and the order in which they occur (and, if available, prosody etc.).

Context: A set of atomic and ground context atoms describing a part of the world.

Phrases: A set of atomic and ground formulas giving the phrases contained in the *Discourse* that are grammatically correct and consistent with *Context*.

Grammar: A set of formulas for the form

$$\mathit{Constituents} \land \mathit{Facts} \rightarrow \mathit{Phrase},$$

where *Constituents* and *Phrase* are sets of grammar atoms, *Facts* a set of context atoms.

We require the following fundamental relation referred to as *faithfulness* between the components:

$$Grammar \wedge Context \wedge Discourse \rightarrow Phrases \tag{3}$$

This means that the *Discourse* and the *Phrases* in it are true to the *Context* and correctly formulated with respect to the *Grammar*.

In case of an ambiguous grammar, we can expect different interpretations for different parses of the string and there may also be different interpretations due to alternative choices of facts from the context.

We do not require the grammar to be unambiguous, but assume a criterion of *unambiguity* of a pair $\langle Phrases, Context\rangle$ which is particular to the grammar formalism applied and not specified further in the general formulation. An unambiguous pair $\langle Phrases, Context\rangle$ is though of as one syntactic and semantic analysis of a *Discourse*. Not every unambiguous such pair is interesting:

Definition 1. *An unambiguous pair* $\langle Phrases, Context\rangle$ *is a* competent interpretation *of given Discourse with respect to given Grammar whenever faithfulness and the following conditions hold:*

1. *(Minimality of Context) If any element is removed from Context, faithfulness fails to hold.*
2. *(Maximality of Phrases) If any new element is added to Phrases, unambiguity or faithfulness fails to hold.*
3. *(Analysis is exhaustive) No new elements can be added to Context which allow an extension of Phrases so that points 1 and 2, and faithfulness are preserved.*

A language interpretation problem *is a problem, given Grammar and Discourse of finding a competent interpretation.*

The condition of exhaustive interpretation excludes $Context = Phrases = \emptyset$ unless the *Discourse* is completely senseless. No negation is possible at the present stage, so any notion of consistency for *Context* needs to be described at the meta-level as part of unambiguity. An example of an unambiguity condition and the implied competence criterion is given section 4.2 for the CHRG grammar formalism [8].

Language interpretation is partly deductive and partly abductive: The *Context* is a premise in (3) and by standard usage, the finding of it is an abductive problem. Identifying phrases is a mainly deductive parsing process, applying grammar rules over and over, however, interacting with abduction in order to have the necessary contextual facts ready.

The following fundamental theorem shows how a language interpretation problem can be reformulated in purely deductive terms, i.e., by an expression with no unknown premises.

Theorem 1. *Let Discourse, Context, Phrases, and Grammar be as above. Then faithfulness (3) with minimality of Context is equivalent to*

$$\mathit{Grammar}' \wedge \mathit{Discourse} \rightarrow \mathit{Context} \wedge \mathit{Phrases} \tag{4}$$

where Grammar′ consists of, for each rule $(\mathit{Constituents} \wedge \mathit{Facts} \rightarrow \mathit{Phrase}) \in$ *Grammar, the following:*

$$\mathit{Constituents} \rightarrow \mathit{Facts} \wedge \mathit{Phrase} \tag{5}$$

Minimality of context is essential for this equivalence as the transformed grammar rules have no way to deduce facts that are not embedded the given *Discourse*.

Proof. Define G to be the minimal set of ground instances of rules in *Grammar* so that $\text{closure}(G \wedge \mathit{Context} \wedge \mathit{Discourse}) = \text{closure}(\mathit{Grammar} \wedge \mathit{Context} \wedge \mathit{Discourse})$, and G' the set of rules constructed from G by changing each $C \wedge F \rightarrow P$ into $C \rightarrow F \wedge P$. Finally, let G_0 be the set of rules constructed from G by removing the *Facts* part, i.e., changing $C \wedge F \rightarrow P$ into $C \rightarrow P$.

We have that (3) is equivalent to

$$G \wedge \mathit{Context} \wedge \mathit{Discourse} \rightarrow \mathit{Phrases} \tag{6}$$

which, by construction of G_0, is equivalent to

$$G_0 \wedge \mathit{Discourse} \rightarrow \mathit{Phrases} \tag{7}$$

I.e., we have eliminated *Context* by using a specialized grammar. The rules of G' differs from those of G_0 by introducing on the righthand side an element of *Context* and referring to minimality of *Context*, we have that

$$\text{closure}(G' \wedge \mathit{Discourse}) = \text{closure}(G_0 \wedge \mathit{Discourse}) \cup \mathit{Context}. \tag{8}$$

Thus (7) is equivalent to

$$G' \wedge \mathit{Discourse} \rightarrow \mathit{Context} \wedge \mathit{Phrases} \tag{9}$$

which obviously is equivalent to (4). QED

3 Adding background knowledge

Background knowledge is a theory that is assumed always to be added to the actual *Context*, and we should be precise to state that *Context* refers to the set of facts that are specific to a given *Discourse*. Here we describe how a background theory in an abductive language interpretation problem can be transformed into a deductive version that fits into a generalization of theorem 1; the transformation is similar to an approach to abduction in CHR described by [1].

Let us mention as *semantic* the predicates that can appear in a *Context* and a background theory *Background*. We distinguish semantic predicates in classes of *abducible* and *defined* predicates and also the *built-in* predicates "$=$" and "$\neq$" with their usual meaning of syntactic equality and nonequality, `false` for falsity, and perhaps others. A *semantic literal* is either an atom of a defined or built-in predicate, or a perhaps negated atom of an abducible predicate.

A *Context* consists of ground abducible literals, and the *Background* theory may contain the following kinds of formulas:

– Ground atoms of abducible literals.
– *Integrity constraints* of the form $A \rightarrow B$ where A is a conjunction of abducible literals, B a conjunction of abducible and built-in literals.
– *Definitions* of the form $S \rightarrow p(\bar{t})$ where p is a defined predicate and S a conjunction of semantic literals.

The set of definitions in a background theory is taken as an abbreviation for their Clark completion.

The faithfulness condition is now the following

$$\mathit{Grammar} \wedge \mathit{Background} \wedge \mathit{Context} \wedge \mathit{Discourse} \rightarrow \mathit{Phrases} \tag{10}$$

with the extension that $\mathit{Grammar} \wedge \mathit{Background} \wedge \mathit{Context}$ must be consistent.

For a theory *Background*, we construct another theory $\mathit{Background}'$ in the following way. Negation in $\mathit{Background}'$ is treated as explicit negation so that the negation symbol "$\neg$" is considered a letter, i.e., "$\neg p$" is a predicate symbol written with two letters, and with "$\neg\neg p$" read as "p".

For each defined predicate p with a set of definitions in *Background*,

$$S_1 \rightarrow p(\bar{t_1}), \ldots, S_n \rightarrow p(\bar{t_n}), \tag{11}$$

$\mathit{Background}'$ includes

$$p(\bar{x}) \leftrightarrow (\bar{x} = \bar{t_1} \wedge S_1) \;\vee\; \cdots \;\vee\; (\bar{x} = \bar{t_n} \wedge S_n)) \tag{12}$$

where the variables $\bar{x}$ do not occur in the original definitions for p.

Integrity constraints in *Background* are included as they are in $\mathit{Background}'$, and for each abducible predicate a, $\mathit{Background}'$ includes

$$a(\bar{x}) \wedge \neg a(\bar{x}) \rightarrow \texttt{false}. \tag{13}$$

Abducible literals contained in *Background* are included directly in $\mathit{Background}'$.

We have the following generalization of theorem 1 giving a deductive version of abductive language interpretation with background knowledge.

Theorem 2. *Let Discourse, Context, Phrases, Grammar, Grammar$'$, Background, and Background$'$ be as above. Then faithfulness (10) with minimality of Context is equivalent to*

$$\mathit{Grammar}' \wedge \mathit{Background}' \wedge \mathit{Discourse} \rightarrow \mathit{Context} \wedge \mathit{Phrases}. \tag{14}$$

Proof. Analogous to the proof of theorem 1.

4 Language interpretation with Constraint Handling Rules

The transformation described above of a language interpretation problem into a deduction form leads to implication formulas with conjunctions in the conclusions and with no obvious way to use Prolog as the underlying engine. However, such formulas fit into the language of Constraint Handling Rules [17] (CHR) and can be executed directly as bottom-up rewriting rules. CHR works on constraint stores that are sets of first order atoms. A *simplification rule* of CHR takes the form *Head* `<=>` *Guard* `|` *Body*. Its logical meaning is (with quantifiers left out) *Guard* → (*Head* ↔ *Body*) and operationally, if constraints matching its *Head* so that *Guard* holds are found in the constraint store, these constraints are replaced by the relevant instance of those given by *Body*. A *propagation rule* is similar but with the arrow "`==>`". Its logical meaning is given by an implication and operationally it works as a simplification except that no constraints are removed from the store. A third kind of rules called *simpagation* is a mixture between the two. For a CHR program P of simplification rules only, the operational semantics produces a representation of closure(P) as final state, and in the general case the relation is more complicated; we refer to [17] for a full presentation.

4.1 Logical grammars in CHR

Logical grammars can be expressed in a straightforward way in CHR. We use here a notation called CHR Grammars [8] (CHRGs) that can be understood as syntactic sugar over CHR rules in exactly the same way as Definite Clause Grammars [22] (DCGs) relate to Prolog. CHRG has counterparts to CHR's simplification and propagation rules written with arrows "`<:>`" and "`::>`". A grammar can refer to grammar symbols as well as constraints in the usual CHR sense. The notation includes a generalized form of look-ahead as *left* and *right grammatical contexts* that provide a high degree of flexibility to characterize linguistic phenomena; see the CHRG web site [7] for source code, examples, and related papers.

We introduce the notation by a prototypical propagation rule preceded by declarations of its constraints and grammar symbols. The notation is inspired by DCG with square brackets for terminal symbols and curly brackets to indicate nongrammatical material.

```
constraints h/1.
grammar_symbols a/0, b/1, d/1, e/2.
a -\ b(X), [c], {h(Y)} /- d(Y) ::> e(X,Y), {h(X)}.
```

It is understood as the following CHR declaration and rule:

```
constraints h/1, a/2, b/3, d/3, e/4, token/3.
a(N0,N1), b(N1,N2,X), token(N2,N3,c), h(Y), d(N3,N4,Y)
  ==> e(N1,N3,X,Y), h(X).
```

The extra arguments added to grammar symbols correspond to markers in the input string, and an input sequence such as "the man walks" is entered as `token(0,1,the)`, `token(1,2,man)`, `token(2,3,walks)`. The grammar rule above says intuitively that a sequence `b(`x`)`, `[c]` is recognized as an `e(`x`,`y`)` giving new constraint `h(`x`)` provided that there is a constraint `h(`y`)` and that the indicated left and right contexts are present. Notice that the order of left and right hand sides is opposite that of DCG. When a terminal string is entered into the constraint store, the rules work as a bottom-up parser giving all possible parses of the string including alternative parses for the same string in case of ambiguity.

For grammars of propagation rules only, syntactic derivation can be defined in a top-down manner in the usual way and with respect to which the parsing provided can be shown to be correct. When simplification rules and hypotheses are included, the bottom-up derivations need to be taken as definition.

The context parts allow tagger-like grammar rules [3], they can be used for disambiguating simple and ambiguous context-free grammar rules, and provide also a way to handle coordination problems in natural language; see [8, 7] for a full introduction.

4.2 Representing language interpretation problems in CHR

Consider a language interpretation problem with background knowledge as described in sections 2 and 3 and with CHRG as the underlying grammar formalism. Grammar symbols, abducible and defined predicates are declared as constraints. Background theories can be represented in two ways, either as rules of CHR since the translated formulas of the form (12) can be understood as simplification rules, or, more efficiently, taking the original clauses as Prolog rules employing the fact that CHR is an extension to Prolog. Disjunctions on the righthand sides (or different Prolog rules for a given predicate) may lead to backtracking during execution.

Built-in "=" atoms are executed correctly by Prolog, and "$\neq$" can be replaced by SICStus Prolog's `dif/2` predicate, that delays until the arguments are sufficiently instantiated to tell them either identical or nonunifiable. Integrity constraints, including those extra added for explicit negation (13) go directly into CHR as propagation rules.

Grammar rules that refer to the semantic *Context* is expected to be written directly in their deductive form, i.e., writing ($\mathit{Constituents} \wedge \mathit{Facts} \rightarrow \mathit{Phrase}$) as *Constituents* `::>` $\mathit{Facts} \wedge \mathit{Phrase}$, and analogously for rules with grammatical context parts and for simplification rules.

To process a language interpretation problem, the input string *Discourse* is entered as a set of `token` constraints and the indicated collection of rules performs a bottom-up parsing, producing abducibles when grammar rules apply, perhaps indirectly when defined predicates are involved; integrity constraints may be triggered whenever an abducible is created or affected by a unification. The resulting interpretation $\langle \mathit{Phrases}, \mathit{Context} \rangle$ can be read out of the final state, however also counting those constraints that have been removed at some stage by a simplification rule; different possible interpretations are given by backtracking. The *Context* generated needs not be minimal in case there are different ways to prove the same goal, but it is minimal with respect to the actual proof.

In order to compare language interpretation in CHR with our theory, we must also give an criterion for unambiguity. We can do with a general version as background theories are capable of expressing a variety of integrity constraints that characterize a given application area. We define an interpretation $\langle$*Phrases*, *Context*$\rangle$ to be *unambiguous* provided

– *Context* $\wedge$ *Background* is consistent.
– For any two grammar symbols `p(`i`,`j`,`$\cdots$`)`, `q(`k`,`ℓ`,`$\cdots$`)` $\in$ *Phrases* $\cup$ *Discourse* it holds that
  - if $i \leq k < j \leq \ell$, then $i = k$ and $j = \ell$, and
  - if $i \leq k \leq \ell \leq j$, then q($k$,$\ell$,$\cdots$) corresponds to a subtree of `p(`i`,`j`,`$\cdots$`)` or the other way round; trees and subtrees are defined in the usual way.

The consistency of *Context* $\wedge$ *Background* is granted provided that *Background* is consistent, otherwise the integrity constraints in *Background* would have produced a failure. Syntactic unambiguity is given not as a property of the grammar but as a property of a particular parse. If the grammar in itself is locally unambiguous — perhaps obtained by proper use of syntactic contexts, extra hypotheses, and procedural thinking — any interpretation will be unambiguous. Local unambiguity is a stronger property than unambiguity, basically telling that the condition above always holds. In case of an ambiguous grammar, extra machinery needs to be added so that the mentioned, syntactic restrictions are maintained during the process and alternative parses generated under backtracking or in parallel. For simplicity, we consider only syntactically unambiguous CHR grammars and compare with the three conditions for competent interpretation of definition 1.

Minimality of *Context*: As already mentioned, the *Context* found may be slightly larger than minimal due the well-known problem with alternative proofs.

Maximality of *Phrases*: Obvious; given by the correctness of the applied parsing strategy.

Analysis is exhaustive: Have any *Context* facts been ignored so that the parser could have continued if it had been aware of them? This is not possible due to theorem 2 and that CHR's procedural semantics ensures that any rule that can apply is applied.

4.3 Compacting the solutions

The final state may include abducible atoms with variables with the meaning that any ground assignment to such variables (not conflicting with integrity constraints) represents a solution to the abductive problem. Consider as an example the following set of abducible atoms returned as answer {`h(a)`, `h(X)`, `h(Y)`}. It may subsume solutions with `X=Y`, `X=Y=a`, or maybe just one of the variables equal to `a`, whatever may be consistent with the integrity constraints. So we may have that `X=b` and `Y=c` give a solution {`h(a)`, `h(b)`, `h(c)`} and `X=a` and `Y=a` another {`h(a)`}; both may be minimal but there may be reasons to prefer the one with fewest elements.

It is possible to extend our method so that it dynamically tries to compact solutions by equating new abducibles to existing ones as a first choice, and then generate the other possibilities under backtracking.

To provide this, we may add for each abducible predicate, two propagation rules, here shown for a predicate `h` of arity one.

```
h(X), h(Y) ==> (X=Y ; dif(X,Y))                (15)

¬h(X), ¬h(Y) ==> (X=Y ; dif(X,Y))              (16)
```

Whenever a new abducible fact, say `h(a)` or `h(X)`, is created by the application of some rule, (15) is applied provided there is another fact `p(`*t*`)` in the constraint store.

An optimization of (15) using facilities of the implemented version of CHR (see [23] for details) is in place, and analogously for (16):

```
h(X), h(Y)#Id ==> (\+X==Y, unifiable(X,Y)) | (X=Y ; dif(X,Y))
pragma passive(Id)
                                                               (17)
```

The `pragma` prevents the rule from being activated twice due to the symmetry in its head and the purpose of the guard is to suppress useless applications.

However, in many cases the problem does not exist since strong user-defined integrity constraints may instantiate and equate abducibles sufficiently during the computation; this is the case in the example below, section 5. The implemented system [7] includes the compaction principle as an option.

5 An example

In the following we show an example of a grammar and background theory formulated in CHR extended with the CHRG notation. We consider language interpretation of discourses such as the following.

Garfield eats Mickey, Tom eats Jerry, Jerry is mouse,
Tom is cat, Mickey is mouse. (18)

What we intend to learn from (18) are the categories to which the mentioned proper names belong and which categories that are food items for others. An interesting question is to which category Garfield belongs as this is not mentioned explicitly. We define the following vocabulary; the `abducibles` declaration is synonymous with `constraints` except that it also introduces predicates for negated abducibles and the integrity constraint (13) that implement negation.

```
abducibles food_for/2, categ_of/2.
grammar_symbols name/1, verb/1, sentence/1, category/1.
```

The background theory is the following consisting of integrity constraints only.

```
categ_of(N,C1), categ_of(N,C2) ==> C1=C2.
food_for(C1,C), food_for(C2,C) ==> C1=C2.
```

I.e., the category for a name is unique, and for the sake of this example it is assumed that a given category is the food item for at most one other category. The following part of the grammar classifies the different tokens.

```
[tom] ::> name(tom).
...
[is]  ::> verb(is).
...
verb(is) -\ [X] <:> category(X).
```

The last rule applies a syntactic left context part in order to classify any symbol to the right of an occurrence of "`is`" as a `category`.

A sentence such as "Tom is cat" is only faithful to a context if `categ_of(tom, cat)` holds in it. So the grammar in the original specification of the current language interpretation problem may contain the following rule.

$$\mathit{name}(i_1,\, i_2,\, N) \wedge \mathit{verb}(i_2,\, i_3,\, is) \wedge \mathit{category}(i_3,\, i_4,\, C) \wedge \mathit{categ\text{-}of}(N,C) \\ \rightarrow \mathit{sentence}(is(N,C)) \qquad (19)$$

By moving the context condition from the premises to the conclusion we achieve a rule that can contribute to solve the problem deductively. In CHRG it becomes the following:

```
name(N), verb(is), category(C) ::>
   {categ_of(N,C)},
   sentence(is(N,C)).
```

A sentence such as "Tom eats Jerry" is only faithful to a context if the proper `categ_of` and `food_for` facts hold in it. A CHRG rule with this in its conclusion looks as follows.

```
name(N1), verb(eats), name(N2) ::>
    {categ_of(N1,C1), categ_of(N2,C2), food_for(C1,C2)},
    sentence(eats(N1,N2)).
```

Let us now trace the processing of the discourse (18) when entered into the constraint store; we record only the context facts. "Garfield eats Mickey" gives rise to

```
categ_of(garfield,X1), categ_of(mickey,X2), food_for(X1,X2).
```

The "X"s are uninstantiated variables. The next "Tom eats Jerry" gives

```
categ_of(tom,X3), categ_of(jerry,X4), food_for(X3,X4).
```

"Jerry is mouse" gives `categ_of(jerry,mouse)`, and the background theory immediately unifies `X4` with `mouse`. In a similar way "Tom is cat" gives rise to a unification of `X3` with `cat` and `food_for(X3,X4)` has become

```
food_for(cat,mouse).
```

Finally "Mickey is mouse" produces `categ_of(mickey,mouse)` that triggers the first integrity constraint unifying `X2` with `mouse` and thus the second integrity constraint sets `X1=cat` and there is no other possibility. So as part of the solution to this language interpretation problem, we have found that Garfield is a cat.

6 Related work

6.1 Abduction for language interpretation

The advantages of abduction for language interpretation — as theoretical model or as implementation — has been recognized be several authors, e.g., [5, 18, 19, 6] just to mention a small fraction, and this is taken in the present work as an established fact.

Purely deductive interpretation as is the only possible way when using, say, a conventional implementation of DCGs [22], is a process of synthesizing the meaning of a phrase from the meanings of its subphrases. This works well when context is known and every piece of information to be extracted is expressed in an explicit way. Abduction is in favour for more subtle meanings given, e.g, by linguistic implicature, and when the attention is on context comprehension. In [10] we have related Stallnaker's [24] view of context comprehension with abductive language interpretation.

A standard objection against abduction for language interpretation is its inherent high complexity, but we prefer to explain this as a property of the very problem that is approached. For practical purposes, the hypotheses to be abduced should be held as general as possible, perhaps including representation by means of disjunction constraints. Carefully designed integrity constraints are also useful. Priorities and pruning based on local priorities are also relevant, but such practically relevant matters are not included in the logical formalization shown in the present paper. Another important issue is possible restrictions in the scope of different hypothesis that we discuss in section 7.

6.2 Approaches to abduction

It is interesting to compare the way we implement abduction with other logically based methods. A remarkable property of our approach is that no meta-level overhead exists. The abductive problem of language interpretation is translated into a purely deductive problem concerning the same predicates and it can be solved by existing technology for bottom-up deduction such as CHR. In this way, our method will take advantage of any future improvements of such technology. We put an emphasis on work that relates abduction with deduction; for a more detailed overview of the field, we refer to [16, 21].

It has been shown by Console et al. [12] how a class of abductive problems can be solved by deductive reasoning in the completion of the definitions; however, no integrity constraints were considered in this early work and no actual implementation was demonstrated. The basic idea is to add the observation to the completion and the explanation in terms of abducibles falls out deductively. As a simple example, assume one definition $h \to p$ where h is abducible. For the observation p we reason in the theory $(h \leftrightarrow p) \wedge p$ and the explanation h follows immediately. We can adapt their ideas to the language interpretation problem which we illustrate by an example. Consider a small grammar with terminal symbols a, b, nonterminal c, and h an abducible describing a context in which the sequence ab can be meaningfully put together to form a c, thus the rule $a \wedge b \wedge h \to c$. For a given, discourse "a, b", we add a and b as axioms, and we reason within $(a \wedge b \wedge h \leftrightarrow c) \wedge a \wedge b$. This gives $h \leftrightarrow c$, in other words, if we chose to believe that c is an acceptable phrase read from "a, b", we have to believe also h.

The relation between abduction and negation by failure has been pointed out by Eshghi and Kowalski [15]. The essence can be illustrated by an example. Assume abducibles h_i and definitions $h_1 \to p$, $h_2 \wedge q \to p$, $h_3 \wedge r \to p$, and $h_4 \to q$. In order to find an explanation for observation p, we consider the consequences of $\neg p$ found by negation as failure. Comparing $\neg p$ with the definitions for p we get $\neg h_1 \wedge (\neg h_2 \vee \neg q) \wedge (\neg h_3 \vee \neg r)$. For $\neg q$ to hold we must have $\neg h_4$, and since $\neg r$ is the case, we get next $\neg h_1 \wedge (\neg h_2 \vee \neg h_4)$. The possible explanations are given by the negation of this, i.e., $h_1 \vee (h_2 \wedge h_4)$. A meta-level procedure based on this principle could be used for top-down abductive analysis based on definite clause grammars.

Most published algorithms for abduction work top-down more or less as a Prolog interpreter with an extra facility to add those abducible whose absence otherwise would make the computation fail, and typically interacting with a side-algorithm to check integrity constraints. We refer to [21] for an overview of this field and mention also our previous work using a reversible metainterpreter for abduction [9, 11].

A paper of Bry [4] says in its title "abduction as deduction" referring to an algorithm encoded as a meta-interpreter that is executed deductively as a Prolog program. Interestingly, this approach is based on model generation that resembles the bottom-up derivations used in the present paper. Other interesting approaches to abduction involving model generation are given by Denecker and De Schreye [14] and Inoue *et al* [20].

The CHR language [17] was introduced by Frühwirth as a tool for writing constraint solvers in a declarative way for traditional constraint domains such as real or integer numbers and finite domains. CHR has proved to be of more general interest and is available as extension of, among others, SICStus Prolog [23]. Being of special interest to language processing, Abdennadher and Schütz [2] has shown that CHR adds bottom-up evaluation to Prolog and a flexibility to combine top-down and bottom-up; Abdennadher and Christiansen [1] has taken this a step further showing that top-down evaluation of abductive logic programs can be expressed directly in CHR in a top-down manner. This principle is applied in section 3 for the encoding of background knowledge.

The bottom-up evaluation of modified grammar rules for abduction that we use seems to be new, turning $a \wedge b \wedge h \rightarrow c$ into $a \wedge b \rightarrow h \wedge c$ which at first a first glance seems to be a logical beginner's mistake.

6.3 A final note on abduction and Definite Clause Grammars

As a side effect of the present work, we have identified how the method of Abdennadher and Christiansen [1] for abduction in CHR, can be modified to a method that runs in Prolog plus CHR, where CHR takes care of abducibles and integrity constraints, and any clause in the background theory is executed by Prolog. As soon as Prolog enters a call to an abducible predicate, CHR will take action by adding the abducible to the constraints store, thus triggering possible integrity constraints; following this, control goes back to Prolog that continues its execution or backtracks depending on the result of the previous; technical details are easily extracted from section 4.2.

This principle works also in case the Prolog program at hand is a Definite Clause Grammar, which can be used with no extra level of translation or interpretational overhead; only when an abducible shows up, execution steps away from Prolog's efficient top-down procedure. No comparisons have been made, but this method seems to be highly competitive in speed compared with any other abductive procedure for logic programs applied to language interpretation, including the one which is the primary focus of the present paper.

7 Conclusion and future work

A translation has been formalized and proved correct of abductive language interpretation problems into a deductive form that can be solved by bottom-up evaluation. We have shown how this can be expressed and evaluated in the language of Constraint Handling Rules (CHR).

The general model covers ambiguous grammars, but the implementation in CHR described here requires unambiguous parsing. It has not been worked out in detail, but it seems possible to add extra constraints to prevent ambiguous grammars from performing steps that involve a given subtree in two unrelated trees. Different parses are then generated under backtracking. Backtracking may also occur in case a background theory has more than one definition for a predicate and when different hypotheses can apply in a grammar rule. An ideal solution seems to be to avoid backtracking and instead evaluate all alternative parses and hypotheses sets in parallel with sharing applied whenever possible. — This is on the agenda for future work.

A critical remark to abductive language interpretation as we (and most other authors in the field) have defined it, is that the *Context* component is one unstructured set that is supposed to go for the whole discourse. When abduction is used for anaphora we simply get too many solution. In most cases, a pronoun such as "he" refers to a character already introduced in the discourse and not to someone appearing twenty pages later. Context may also change during the discourse, e.g., due to a psychological development of the characters as the story goes on. We see a need for structuring the *Context* and to have ways of indication scope of hypotheses. Assumption Grammars [13] give one suggestion for such scoping mechanisms: Hypotheses are explicitly created and explicitly applied and it may be specified that creation comes before application and whether a hypothesis is supposed to be used once or several times. This still seems too primitive for a general approach to abductive language interpretation with dynamic or scoped context but it can give inspiration. An implementation of Assumption Grammars in CHR in a way quite similar to the present approach to abduction can be found in [7].

Another relevant criticism of the present work raised by a reviewer (which actually can apply to the different works on abductive language interpretation that we are aware of) is the representation of context as a set of atomic statements. To provide a model closer to the human way of comprehension, we should allow also the possible learning of rules from a given text, e.g., due to recurring patterns. In other words, text comprehension should also involve an element of induction. The method we have applied and other approaches to abduction in language interpretation have no immediate generalization to include induction, but the biography [16] on the relation between abduction and induction may give inspiration for extensions in this direction.

Acknowledgment: This research is supported in part by the OntoQuery funded by the Danish Research Councils, and the IT-University of Copenhagen.

References

1. Abdennadher, S., Christiansen, H., An Experimental CLP Platform for Integrity Constraints and Abduction. Proceedings of FQAS2000, Flexible Query Answering Systems, pp. 141–152, *Advances in Soft Computing series,* Physica-Verlag (Springer), 2000.
2. Abdennadher, S., Schütz, H. CHR$^\vee$: A flexible query language. Proc. FQAS'98, *Lecture Notes in Artificial Intelligence* 1495, pp. 1–14, Springer, 1998.
3. Brill, E., Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics* 21(4), p. 543–565, 1995.
4. Bry, F., Intensional Updates: Abduction via Deduction. *Logic Programming, Proceedings of the Seventh International Conference*, pp. 561–575, MIT Press, 1990.
5. Charniak, E., McDermott, D., *Introduction to Artificial Intelligence.* Addison-Wesley, 1985.
6. Christiansen, H., Why should grammars not adapt themselves to context and discourse? *4th International Pragmatics Conference, Kobe, Japan, July 23–30 1993*, (Abstract collection), International Pragmatics Association p. 23, 1993. (Extended abstract: http://www.dat.ruc.dk/~henning/IPRA93.ps).
7. Christiansen, H., *CHR Grammar web site*, http://www.dat.ruc.dk/~henning/chrg/

8. Christiansen, H., Logical grammars based on constraint handling rules, (Poster abstract). *Proc. Int'l Conference on Logic Programming*, Lecture Notes in Computer Science (to appear), Springer 2002.
9. Christiansen, H., Automated reasoning with a constraint-based metainterpreter. *Journal of Logic Programming*, 37(1-3):213–254, 1998.
10. Christiansen, H., Open theories and abduction for context and accommodation. 2nd International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT'99) Bouquet, P., Brezillon, P., Serafini, L. (eds.) *Lecture Notes in Artificial Intelligence* 1688. Springer-Verlag, pp. 455–458, 1999.
11. Christiansen, H., Abduction and induction combined in a metalogic framework. *Abductive and Inductive Reasoning: Essays on their Relation and Integration*, Flach. P., Kakas, A., (eds.), Kluwer Academic Publishers. pp. 195–211, 2000.
12. Console, L., Theseider Dupré, D., Torasso, P., On the Relationship between Abduction and Deduction. *Journal of Logic and Computation* 1(5), pp. 661–690, 1991.
13. Dahl, V., Tarau, P., Li, R., Assumption grammars for processing natural language. *Proc. Fourteenth International Conference on Logic Programming.* pp. 256–270, MIT Press, 1997.
14. Denecker, M., De Schreye, D., On the Duality of Abduction and Model Generation. *Proc. of the International Conference on Fifth Generation Computer Systems 1992*, pp. 650–657, 1992.
15. Eshghi, K., Kowalski, R.A., Abduction Compared with Negation by Failure. *Logic Programming, Proceedings of the Sixth International Conference*, MIT Press pp. 234–254, 1989.
16. Flach. P., Kakas, A., (eds.), *Abductive and Inductive Reasoning: Essays on their Relation and Integration*, Kluwer Academic Publishers. pp. 195–211, 2000.
17. Frühwirth, T.W., Theory and Practice of Constraint Handling Rules, *Journal of Logic Programming*, Vol. 37(1–3), pp. 95–138, 1998.
18. Gabbay, D., Kempson, R., Pitt, J., Labeled abduction and relevance reasoning. *Nonstandard Queries and Nonstandard Answers*, Demolombe, R., Imielinski, T., (eds.), pp. 155–185, Oxford Science Publications, 1994.
19. Hobbs, J.R., Stickel, M.E., Appelt D.E., and Martin, P., Interpretation as abduction. *Artificial Intelligence* 63, pp. 69-142, 1993.
20. Inoue, K., Ohta, Y., Hasegawa, R., Nakashima, M., Bottom-up Abduction by Model generation. *Proc. of IJCAI-93, Proceedings of the Thirteenth International Conference on Artificial Intelligence*, pp. 102–108, 1993.
21. Kakas, A.A., Kowalski, R.A., Toni, F., The role of abduction in logic programming, *Handbook of Logic in Artificial Intelligence and Logic Programming*, vol. 5, Gabbay, D.M, Hogger, C.J., Robinson, J.A., (eds.), Oxford University Press, pp. 235-324, 1998.
22. Pereira, F.C.N., Warren, D.H.D., Definite clause grammars for language analysis. A survey of the formalism and a comparison with augmented transition grammars. *Artificial Intelligence* 10, no. 3–4, pp. 165–176, 1980.
23. *SICStus Prolog user's manual.* Version 3.8, SICS, Swedish Institute of Computer Science, 2001. Most recent version available at `http://www.sics.se/isl`.
24. Stalnaker, R., On the representation of context. *Journal of Logic, Language, and Information*, vol. 7, pp. 3–19, 1998.