REVERSIBLE RESOLUTION WITH AN APPLICATION TO PARAPHRASING

Matthew Hurst

Department of Computer Science, The University of Sheffield, email: M.Hurst@dcs.shef.ac.uk

This paper describes a reversible resolution method based on proof procedures. A monotonic semantics is given for a subset of the Core Language Engine (CLE)'s quasi-logical form *(QLF) [AC92] which defines a relation (CAT) realised by a series of declarative statements describing possible resolutions for terms. The application of paraphrasing is used to demonstrate the method in a useful function in a Natural Language Processing environment.*

I would like to acknowledge the assistance of Dick Crouch of SRI International.

1 RESOLUTION

The topic of this paper is the resolution of inter-sentential referential terms. The work presented extends to general resolution of partial analysis under the monotonic interpretation paradigm of the CLE.

1.1 Reference Resolution

Generally, referential resolution is the process by which the surface analysis of a term in a sentence is in some way completed to become associated with a context of discourse. This process involves association with a preceding phrase or discourse entity.

A mechanism capable of *reversible resolution* takes an analysis and associates it with some entity in and with respect to some context (in the **forward** direction); it is also capable of taking some entity and generating an analysis (and ultimately a natural language string) in and with respect to some context (the **backward** direction).

1.2 Resolution and Monotonic Interpretation

The core of the work in this paper is an extension of the semantics of the **QLF** formalism used by the CLE. The extension is significant in that it provides a monotonic semantics for resolution, described as a relation that is stated declaratively as a set of *resolution rules*. In being declarative, these rules can be used for both interpretation and generation.

The advantages of a declarative treatment of any component of an NLP system are twofold. Firstly, the simplicity of extension and secondly, independence from the underlying implementation. A monotonic interpretation also has desirable qualities in a formal system. The most important of these qualities, for this work, is the reversibility of interpretation/generation. This paper describes a mechanism that gives this functionality to the resolution component of the CLE.

2 THE QLF

The level of representation that this paper deals with is the **QLF**. The main components of which are the *term* and the *formula* (form).

2.1 QLF Syntax

Before continuing with the semantic description, a brief summary of the *abstract* syntax of the **QLF** term is given.

The structure of the term that shall be focussed on (`term(...)`) has five major components.

- The *index* (I). This is a unique identifier associated with a particular `term`.
- The *category* (C). The linguistic category of the expression: a list of feature value tuples.
- The *restriction* (R). A first order one place predicate describing the term.
- The *quantifier*. A generalised quantifier, i.e. a cardinality predicate holding of two properties.
- The *referent*. An expression of reference, either a constant or a term index.

Terms may also be variables, indices or constants. The QLF formula is similar in structure to the term (`form(...)` is defined in [AC92]).

Combinations of the fields may be uninstantiated and it is the instantiation of these *meta-variables* that is the effect of resolution. In the case of terms, in the forward direction, the quantifier and referent will be uninstantiated; in the backward direction, the category, restriction and quantifier will be uninstantiated.

2.2 QLF Semantics

The semantics of the **QLF** as presented in [AC92] are extended in the manner described. As shown above, a

QLF may contain any number of *meta-variables* and so the semantics of the language is slightly different from the traditional one. Instead of a function from models to truth values, a *partial* function is defined.

The denotation of a formula F is a partial function [[F]] from models to truth values. This partial function is defined by W, a relation between a formula, a model and a truth value. The quality of a monotonic semantics is apparent in the case when [[F]] with respect to some model is undefined, i.e. $W(\mathrm{F}, m, 1)$ and $W(\mathrm{F}, m, 0)$. This situation occurs when *meta-variables* are present in the QLF and consequentially a *partial interpretation* is being considered. The incremental analysis provides an 'extension' of [[F]] in [[F']] whenever F' is a more resolved version of F.

The semantic rules that discharge *meta-variables* are of the general form: $W(\mathrm{F}, m, v)$ if $W(\mathrm{F}', m, v)$ where F' is an 'extension' of F.

2.3 The Relation $\mathcal{CAT}$

A relation $\mathcal{CAT}$ of category is defined with the following arguments.

- *category*
- *index*
- $\begin{cases} < \textit{referent}, \textit{quantifier} > & \text{if a } \texttt{term}, \\ \textit{property} & \text{if a } \texttt{form} \end{cases}$
- *restriction*
- *context*

Rules describing the discharge of *quantifier* and *referent meta-variables* for `terms` are defined as follows.

1. $W(\mathrm{F}, m, v)$ if $W(\mathrm{F}[\_\mathrm{q}/Q], m, v)$

   The term $\texttt{term}(\mathrm{I}, \mathrm{C}, \mathrm{R}, \_\mathrm{q}, \_\mathrm{r})$ is contained in the formula F

   $\exists Q$: $\mathcal{CAT}(\mathrm{C}, \mathrm{I}, \mathrm{R}, < \_\mathrm{r}, Q >, \mathrm{Ctxt})$

2. $W(\mathrm{F}, m, v)$ if $W(\mathrm{F}[\_\mathrm{r}/REF], m, v)$ and $W(\mathrm{R}(REF), m, 1)$

   The term $\texttt{term}(\mathrm{I}, \mathrm{C}, \mathrm{R}, \_\mathrm{q}, \_\mathrm{r})$ is contained in the formula F

   $\exists REF$: $\mathcal{CAT}(\mathrm{C}, \mathrm{I}, \mathrm{R}, < REF, \_\mathrm{q} >, \mathrm{Ctxt})$

for some *context* `Ctxt`.

What 1 says is that the truth value of F (v) is the truth value of F with, for a particular term, all instances of the quantifier _q replaced by Q. 2 states that the truth value of F is the truth value of F with, for a particular term, all instances of the referent _r replaced by some referent REF; *and* that R, the restriction of that term, when applied to the referent term holds.

A further set of semantic rules are defined to deal with QLF `forms` and quantification.

The complete rule set as defined in [AC92] defines membership of the relation W. It should be noted that the definition found there is underspecified in the semantics for category and the relation $\mathcal{CAT}$ is an expansion of the relation $\mathcal{S}$ (called 'salient').[AC92] mentions that ... *the computational analogue of $\mathcal{S}$ was implemented as a collection of 'resolution rules' in [Als90].* The work reported in this paper and in [Hur93] is a computational analogue of $\mathcal{CAT}$ with a more general and declarative treatment.

2.4 Defining $\mathcal{CAT}$

Given that certain arguments of the $\mathcal{CAT}$ relation are members of infinite sets (*context, restriction*) and that others are dependent on these arguments (*referent*), all elements of $\mathcal{CAT}$ can never be explicitly enumerated. The description of this relation should, then, take the arguments that are finite (*category, quantifier*) and use some compact definition to accommodate the infinite arguments. A definition of $\mathcal{CAT}$ based on theorem proving fulfills this role, as described in the following sections. The declarative definition allows the direction of resolution to be indicated be the *meta-variables* that require instantiation.

3 LOGIC OF RESOLUTION

Before continuing with a more formal definition an example of resolution is presented. The phrase *The girl runs* has the following abstract form.

$\texttt{run}(\texttt{term}(\texttt{idx1}, < \ldots >, \lambda \mathrm{X}.girl(\mathrm{X}), \_\mathrm{q}, \_x))$

Resolving this means instantiating the *meta-variables* _q and _r, and a possible resolution might be to `exists` and `alice` respectively.

We can see how the category limits the selection of the quantifier and the selection of the referent with respect to the restriction, that is - we require a single definite object in context for which the restriction holds. This suggests a definition of $\mathcal{CAT}$ as a series of rules defining how the uninstantiated arguments are formed and how values can be found with respect to the context, for example

$\mathcal{CAT}(\mathrm{C}, \mathrm{I}, \mathrm{R}, < \mathrm{Ent}, \texttt{exists} >, \mathrm{Ctxt})$
$\leftarrow$
single_object_matching_restriction(Ent, R, Ctxt)

Consequently, inference can be used to, in this example, find some value of Ent which holds for R in Ctxt.

3.1 Contextual Entailment of Resolution

The process of resolution in the QLF formalism is simply the instantiation of meta variables with respect to the relation $\mathcal{CAT}$. As described previously, this relation involves the context of processing and can be expressed logically in the following manner.

$\mathrm{Ctxt} \cup \mathrm{Assumption} \models [_{Res}\ \mathrm{F} \Leftrightarrow \mathrm{F}']$

where, for forward resolution ($=>$), F' is a more instantiated version of F; and for backward resolution ($<=$) F is an unresolved QLF of the partially instantiated F'.

The addition of a set of `Assumptions` is included for completeness, though in this work is in fact an empty

set. Assumptions can be used, for example, to promote a possible referent to be most salient when there are two equally likely possibilities.

In brief, resolution of the term *The girl* to some token `alice` in the example would have the entailment

`Ctxt(C)`
$\models$
`run(term(idx1,` $< \ldots >, \lambda X.girl(X), \_q, \_r$`))`
$\Leftrightarrow$
`run(term(idx1,` $< \ldots >, \lambda X.girl(X)$`, exists, alice))`

assuming `alice` is salient.

The conditions on a QLF that lead to equivalence between a resolved and unresolved formula can be encoded as a series of declarative statements. These statements take *category, index, <referent, quantifier>/property, restriction* and implement the interaction of *context* with respect to these by a number of logical primitives. That is to say, the abstract notation used to define $\mathcal{CAT}$ is realised as a rule with either of the following structures: `term(...)` if C; `form(...)` if C where C is a set of logical primitives, principally proof goals, defining $\mathcal{CAT}$, and the `term` and `form` contain the arguments to the relation, context being globally defined.

3.2 Inference for Resolution

If we can transform a QLF into a first order logical representation then we can exploit existing techniques to verify a resolution in context.

Using inference allows resolution to be completely declarative and, therefore, applicable in both directions. All that is required to ensure reversibility is that we allow the inference mechanism to instantiate not only arguments, but the predicates found in restrictions.

A description of the declarative rules will be presented next before continuing with the description of the proof procedure.

3.3 Declarative Rules Defining $\mathcal{CAT}$

One case of resolution is presented showing how the resolution rules employ conditions to define the category semantics for singular definite terms.

Singular Definite Terms

A rule defining the semantics of this category must provide a quantifier and a referent that holds for the relation $\mathcal{CAT}$. The existential quantifier is appropriate in the subset of QLF considered in this work. The referent should be some salient entity such that the category of the coreferent agrees with the category of the term being resolved (i.e. number agreement etc.) *and* that the restriction of the term being resolved holds for that entity.

So we wish to state two conditions that must hold in order for the resolution to be a correct declaration of $\mathcal{CAT}$ for a singular definite term.

The rule, then, is stated as

`term(Idx, [det = the, num = sing], Rstr, exists, ent(Ent))`
$\Leftarrow$
`salient_entity(Ent, Idx, [det = the, num = sing])`
`satisfy_qlf(term(Idx, [det = the, num = sing], Rstr, exists, ent(Ent)))`

4 THEOREM PROVING

This section details the tiered theorem prover that is the core of the resolution mechanism. The axioms associated with each layer are also mentioned.

An intuitive employment of this technique applied to terms as a method of resolution might include the following components, given that we are dealing with a QLF and some conjunctive normal form database.

1. Convert QLF to logical form.

2. Satisfy the logical form with a theorem prover.

However, with the use of suitable axioms we can collapse these into one step. In order to do this, the QLF formalism must be accepted as valid formulae in the language of the theorem prover. The advantage of this method allows a single style of operation to be employed in the satisfaction of QLFs.

In order for this method of resolution to work correctly bi-directionally, it is required that we allow the theorem prover to range over arguments *and* predicates in the database. This allows the algorithm to function in the reverse direction. This being the case, logical terms (after processing) are asserted in the database in a list format, i.e. `[predicate,` $\texttt{arg}_1$`,` $\texttt{arg}_2$`, ...]` instead of `predicate(`$\texttt{arg}_1$`,` $\texttt{arg}_2$`, ...)`. Also, QLF 'templates' (see [Hur93] for a full explanation) are used to provide a desirable set of generatable noun phrase structures in the backward direction.

4.1 Two Layers

There are two layers in the theorem prover. The reason for this is as follows. Ultimately, we want to concern ourselves with proofs like p(`token`). In order to do this we have to transform the original QLF into a series of associated predicates. The arguments to these predicates may be QLFs themselves which require discharge of a quantified term to produce a logical expression. The process of discharge can not be accomplished with a simple declarative rule local to the predicate as a number of representations exist in the CLE QLF formalism that require discharged to a `token` in a procedural manner. Therefore, it is necessary to first perform a number of transformations, together with a procedural implementation of term discharge, as a first stage to using the theorem prover in resolution.

4.2 Top Proof

The first layer of the theorem prover is concerned with QLFs and the QLF *axioms*. The axioms are used to effectively transform the QLF into a first order notation from which point simple inference can be carried out with respect to the facts asserted in the database. This level of proof is straight forward and involves three cases. *Modes ponens, and intro* and the special case, *discharge of terms*.

Case 1 is the use of a QLF axiom. The second case is simply the proof of a conjunct by the proof of its parts. The final case is the discharge of terms. The discharge of terms reduces a `term` expression into a `token` about which inference can be carried out with the assertions in the domain. An algorithm exists to carry out this procedural attachment to the proof [Hur93].

4.3 QLF Axioms

An important component of this level of the proof mechanism is the set of axioms used to represent the relationship between the restriction and the logical language of the database.

The following is an example of an axiom used by the top level of the theorem prover and is rule that encodes the reduction of the restriction of a possessive form.

$$\texttt{form}(\_, \texttt{poss}, \texttt{Index}, \texttt{F}\hat{}[\texttt{and}, [\texttt{P1}, \texttt{A}], [\texttt{F}, \texttt{A}, \texttt{B}]], \texttt{app}(\texttt{X}\hat{}\texttt{Y}\hat{}[\texttt{poss}, \texttt{X}, \texttt{Y}], \texttt{Index})) \Leftarrow [\texttt{poss}, \texttt{A}, \texttt{B}] \wedge [\texttt{P1}, \texttt{A}]$$

which represents the following β reduction.

1. $\lambda F(\texttt{P1}(\texttt{A}) \wedge F(\texttt{A}, \texttt{B})[\lambda X \lambda Y(\texttt{poss}(X, Y))]$
2. $(\texttt{P1}(\texttt{A}) \wedge \lambda X \lambda Y(\texttt{poss}(X, Y))(\texttt{A}, \texttt{B}))$
3. $(\texttt{P1}(\texttt{A}) \wedge \texttt{poss}(\texttt{A}, \texttt{B}))$

which has the same logical form.

4.4 General Theorem Prover

The second layer of the proof procedure takes as input a (possibly partially instantiated) logical expression and an entity and returns, conditionally on success, the fully instantiated logical expression.

There are three different cases which the proof mechanism deals with. These are as follows.

1. A clause is present in the context model which unifies with the *expression* to be shown.
2. An inference rule can be employed to prove the *expression*, in which case each of the antecedent proofs must be constructed.
3. A proof of equality can be constructed, in which case substitution is made between the entities in the equality proof and this equal entity is used in the proof.

4.5 Domain Axioms

The second layer of the theorem prover also has a set of axioms with which to generate proofs. These axioms are either asserted in the database as part of the interaction between the user and the system, or may be explicitly placed there prior to use.

5 PARAPHRASING

The object of paraphrasing is to produce a concise and, if possible, unambiguous description of a previous statement. As this work is only concerned with generation of object descriptions, we want to use the technique described for reversible resolution and produce a string that describes the entity according to a set of evaluation criteria. This set of criteria can then be evaluated in the context of a preference metric used to select the *best* description[1].

Generation is carried out by the CLE. The methods employed to generate from a QLF are discussed in [SvNPM90].

5.1 Constraints on Description

There are a number of constraint that should be considered when producing a noun phrase to describe an object. A suitable subset is:

1. Effort of Realisation (ER): a measure of the effort required to realise the actual words in the medium used.
2. Effort of Association (EA): a measure of the effort required to associate some entity with the description realised.
3. Informational Value (C): a measure of the complexity of the realisation.
4. Domain Coverage (DC): a measure of the number of items in the domain that can be described by the realisation.
5. Effects of Salience: how such things as recency effect the other dimensions.

5.2 Preference by Dimension

The following is a simple interpretation of the above constraints as a first attempt at obtaining a metric for noun phrase preferences.

Each dimension is preferred as 1. Minimise, 2. Minimise, 3. Maximise (with respect to salience), 4. Minimise (unique element of domain preferred); and a simple implementation of the dimensions might be (functions of)

1. ER: the physical length of the orthographic representation.
2. EA: the size of the proof (or a weighted sum of the proof).

[1] Designing a metric for any stage of Natural Language Processing will always have a certain *ad hoc* nature. However, intuitive principals can often guide through the vagaries. Note should be made of Grice's co-operative principle and the maxims that this suggests. Other work in this area, including that of Sperber and Wilson, can be found in [Poz90]

3. C: the number of leaves in the QLF of the resolved form.

4. DC: the set of referents that that noun phrase can be resolved to.

The issue of salience (5) acts (conceptually) by relaxing the ideal constraint that $|DC|$ equals 1. This could operate by either a hard decision involving dividing the set of entities in the domain into those that are salient and those that are not or some manner of grading the salience of an entity.

We can interpret the desired function in the following manner: *The best phrase that uniquely describes the entity.*, and the suboptimal (non unique) result as: *The best phrase that describes the entity* with the minimal domain coverage.

We are still left with the notion of *best phrase*. It is inherently vague and, in the context of the parameter set, is the notion of *most compact*. As the function presented in the next section indicates, this is not necessarily simply the shortest.

5.3 Implementing the Metric

A function that takes the dimensions described and produces a value is as follows :

$$f : ER \times EA \times C \times DC \rightarrow \Omega \cdot \frac{C}{ER \cdot EA \cdot |DC|}$$

where Ω is a weight vector. Maximising this function leads the metric to prefer minimal proofs, strings and domain coverage for the largest complexity (a rough measure of information).

5.3.1 Adding Salience

The more salient an entity is, the more relaxed the constraints on C and the uniqueness condition of DC can become. For example, if ENT1 and ENT2 can be described as *'the boy'*, and ENT1 can be described as *' the boy on the ground'*, assuming equal salience, a suitable noun phrase for ENT1 could be *'the boy on the ground'*. However, if ENT1 is more salient than ENT2 then the shorter phrase, *'the boy'*, might suffice. This intuition modifies the function by S_{ent}, the salience of the entity.

5.3.2 Calculating Salience

Salience is calculated in an *ad hoc* manner using parameters that give some intuitive guide to the natural focus of an entity (e.g. recency, frequency of reference etc.). In this way, an entity is more salient than another if it has been referred to more recently and more frequently. The definition is completed by checking for category agreement.

5.4 Implementing Paraphrasing

Paraphrasing can, then, become a component of term resolution in the forward direction. Finding all possible referents and offering a choice by *paraphrase* provides the users with useful assistance in an interactive session with the NLP system.

5.5 Initial Comparison with CLARE

A simple comparison between the implementation of the method described and the resolution/paraphrasing facilities of CLARE was carried out. The task was to paraphrase for referential ambiguity. The sample text contained two possible referents for the final noun phrase (the player) and is as follows.

The player is on the ground. The player is tall. A player is small. The player runs.

CLARE was incapable of distinguishing the two players with unique paraphrases demonstrating a weakness in its appraisal of ambiguity of noun phrase with respect to context. The system implemented realised that it was necessary to include the adjective in the description of each player. Other tests demonstrated the strength of using templates. For example, CLARE generated the phrase *the boy John is* as a paraphrase of *The boy that likes games.* Such oddities can't occur if templates are incorporated intelligently.

6 CONCLUSION

This paper has demonstrated that a declarative, reversible approach to the problem of resolution is a feasible and desirable feature of a general Natural Language Processing system. It allows the definition of a relationship between category and context which indicates possible reference. Supporting the declarative treatment with a theorem prover also allows certain considerations to be taken into account when ranking possible referents.

References

[AC92] Hiyan Alshawi and Richard Crouch. Monotonic semantic interpretation. In *30th Annual Meeting of The Association for Computational Linguistics*, 1992.

[Als90] Hiyan Alshawi. Resolving quasi logical forms. *Computational Linguistics*, 16(3), September 1990.

[Hur93] Matthew Hurst. Reversible resolution with an application to paraphrasing. Master's thesis, The University of Cambridge, 1993.

[Poz90] Victor Poznański. *A Relevance-Based Utterance Processing System*. PhD thesis, The University of Cambridge, 1990.

[SvNPM90] S. M. Shieber, G. van Noord, F. C. N. Pereira, and R. C. Moore. Semantic head driven generation. *Computational Linguistics*, 16, September 1990.