Order Independent Typed Default Unification

Alex Lascarides
Centre for Cognitive Science *and*
Human Communication Research Centre,
University of Edinburgh,
2, Buccleuch Place,
Edinburgh, EH8 9LW,
Scotland, UK
alex@cogsci.ed.ac.uk

Ann Copestake
Computer Laboratory, University of Cambridge, *and*
Center for the Study of Language and Information,
Stanford University,
Cordura Hall,
Stanford, CA 94305,
USA
aac@csli.stanford.edu

August 19, 1995

Abstract

We describe an order independent typed default unification function, which allows default reentrancies and values on more general types to be overridden by conflicting default information on more specific types. This therefore extends the order independent default unification operations in Young and Rounds (1993) and Lascarides *et al.* (in press), thereby increasing the range of linguistic generalisations that can be expressed.

1 Introduction

Many computational linguists have argued that default inheritance is useful for encoding linguistic generalisations which have exceptions (Dealemans *et al*, 1992). But although there have been several approaches to formalising default inheritance within feature structure languages by defining an operation of default unification (e.g. Bouma 1992, Russell *et al* 1991, Copestake 1993), these have failed to achieve the combination of perspicuity, declarativity and expressibility familiar from unification-based approaches to non-default inheritance. We believe the following properties are necessary for default unification to have these qualities:

1. Non-default information can be distinguished from default information and is always preserved.
2. Default unification behaves like monotonic unification in the cases where monotonic unification would succeed.
3. Default unification returns a single result, deterministically.
4. Defaults can be given a precedence ordering such that more specific information overrides less specific information.
5. Default unification can be described using a binary, order independent (i.e., commutative and associative) operation.
6. Default unification never fails unless there is conflict in non-default information.

Most definitions of default unification have been based on Kaplan's sketch of priority union (Kaplan, 1987) and are asymmetric since one feature structure (FS) is taken to be indefeasible while the other is defeasible. This operation isn't commutative or associative (Carpenter, 1993). Although there are some applications for which an asymmetric operation is useful, the order-dependency makes it undesirable as a basis for inheritance. Furthermore, since these definitions don't meet our criterion 4, the inheritance hierarchy has to be defined separately. The most common linguistic application for default inheritance is to encode lexical generalisations (Boguraev and Pustejovsky, 1990; Briscoe *et al.*, 1990; Vossen and Copestake, 1993; Daelemans 1987; Evans and Gazdar 1989a,b; Flickinger *et al.*, 1985; Flickinger, 1987; Flickinger and Nerbonne, 1992; Krieger and Nerbonne, 1993; Shieber, 1986 and others). Having an ordering dependent operation in the lexicon itself is not a serious problem, because all the entries to be unified are in a fixed hierarchy and an inheritance order can indeed be imposed. But in a discourse situation, one cannot predict which pieces of information are to be unified *in advance* of starting the discourse parsing process. So the interface between discourse processing and order dependent lexical processing would have to take into account the order in which the unification operations are done, which is impractical (see Lascarides and Copestake 1995 for further discussion).

Young and Rounds define an order independent version of default unification (Young and Rounds, 1993). But it doesn't meet criterion 4. They use Reiter's default logic (Reiter, 1980) to model the operation. So the most straightforward way of extending the definition to meet criterion 4 would be to extend Reiter's default logic so that it validates specificity. But as Asher and Morreau (1991), Lascarides and Asher (1993) and Lascarides *et al* (in press) argue, all such extensions to Reiter's default logic either impose ordering constraints on the application of logical axioms in proofs (e.g., Konolige, 1988), or they impose a context-sensitive translation of the premises into the formal language (e.g,. Brewka, 1991). So extending Young and Rounds' definition in this way comes at the cost of an underlying formal semantics which has separately defined order constraints on the logical axioms, or context sensitive translation.

Lascarides *et al* (in press) use conditional logic to extend Young and Rounds' definition to one on typed feature structures, called Persistent Default Unification (PDU). They use conditional logic precisely because these logics are able to validate nonmonotonic patterns of inference involving specificity without imposing ordering constraints on the application of axioms or imposing context sensitive translation on the premises. In order to allow default inheritance in type hierarchies, precedence of default values in PDU is determined by the specificity of the types at which the default is introduced. PDU thus meets criterion 4 for non-reentrant information. But it fails to validate the overriding of default reentrancy by conflicting default values introduced by a more specific type. In this situation, the default reentrancy survives and the default value on the shared node is $\perp$, indicating unification failure.

This failure to fully meet criterion 4 restricts PDU's linguistic application and also causes problems in interpreting FSs containing $\perp$. For example, consider a representation of the morphology of English verbs where there is an attribute for the past tense form and another for the past particle. Normally these are the same, and this can be represented by default reentrancy. The forms differ in some classes of irregular verbs (e.g. *speak, spoke, spoken*), but the specification of the vowel change and suffixation for this class ought itself to be defeasible, because of a subclass of these verbs, such as *weave*, which are also found with the regular suffixation pattern (*wove/weaved, woven/weaved*).

In this paper, we define an order independent typed default unification operation ($\overset{<>}{\sqcap}$) called YADU (Yet Another Default Unification), which allows default reentrancies to be overridden by default values introduced by a more specific type and thus provides the first definition of default unification that fulfills all of the above criteria.

$$\left(\begin{bmatrix}\text{F} : \mathbf{c} \\ \text{G} : \mathbf{c}\end{bmatrix} \overset{<}{\sqcap} \begin{bmatrix}\text{F} : \mathbf{a} \\ \text{G} : \mathbf{b}\end{bmatrix}\right) \overset{<}{\sqcap} \begin{bmatrix}\text{F} : \boxed{1} \\ \text{G} : \boxed{1}\end{bmatrix} \quad = \quad \begin{bmatrix}\text{F} : \boxed{1}\mathbf{c} \\ \text{G} : \boxed{1}\end{bmatrix}$$

$$\begin{bmatrix}\text{F} : \mathbf{c} \\ \text{G} : \mathbf{c}\end{bmatrix} \overset{<}{\sqcap} \left(\begin{bmatrix}\text{F} : \mathbf{a} \\ \text{G} : \mathbf{b}\end{bmatrix} \overset{<}{\sqcap} \begin{bmatrix}\text{F} : \boxed{1} \\ \text{G} : \boxed{1}\end{bmatrix}\right) \quad = \quad \begin{bmatrix}\text{F} : \mathbf{c} \\ \text{G} : \mathbf{c}\end{bmatrix}$$

Figure 1: Non-associativity of Asymmetric Default Unification

2 Typed Default Feature Structures

We wish to define an order independent default unification operation on typed information. Copestake (1993) and Grover *et al* (1994) define default unification operations that operate over typed feature structures (TFSs). In both cases, the operation is asymmetric, however. The TFS on the left hand side of the operation is treated as indefeasible, and the one on the right hand side is treated as defeasible. Roughly speaking, the result of unifying these two TFSs together is the indefeasible TFS unified with as much information as possible from the defeasible TFS while remaining consistent. An example is shown in Figure 1. Clearly, an asymmetric default unification is non-commmutative, because by changing the order of the arguments, one changes which information is default and which is non-default. The examples in Figure 1 also show that such an operation is non-associative.

Our demand for order independence requires us to define a symmetric operation. But this means that we cannot treat one TFS as specifying wholly indefeasible information and the other as wholly defeasible, as one does in Figure 1. Rather, one must explicitly mark in the information structures which information is to be treated as default and which is non-default. The standard language of TFSs doesn't make such distinctions. But Young and Rounds (1993) and Lascarides *et al.* (in press) demonstrate that it's a necessary prerequisite for defining a symmetric default binary operation. Consequently, we must extend the representation of typed feature structures so that they have the properties essential for successfully defining a default unification operation that meets the criteria we specified in section 1. First, these structures must keep default and non-default information distinct. Second, they must keep track of certain default information that was overridden during unification, and so doesn't appear in the feature structure itself. This is because this information must sometimes influence the results of subsequent default unifications if it is to meet the specified criteria. The purpose of this section is to extend the representation of typed information to go beyond TFSs in the required manner. We do it in a similar way to that deployed in Lascarides *et al* (in press). We will then define the default unification operation over these extended representations.

We refer to these more refined information structures as typed default feature structures or TDFSs. And we use the same notation as Lascarides *et al* (in press) for representing them. Informally, a TDFS contains a 'double' typed feature structure (TFS) in which the left hand FS specifies what is indefeasible and the right-hand FS what is defeasible. For instance in (1), the value of the path F·G is **b** by default, but the existence of the paths F·G and H, and the value **a** for H are non-default.

(1) $$\begin{bmatrix}\mathbf{t} \\ \text{F} : \begin{bmatrix}\text{G} : \top\end{bmatrix} \\ \text{H} : \mathbf{a}\end{bmatrix} \Big/ \begin{bmatrix}\mathbf{t} \\ \text{F} : \begin{bmatrix}\text{G} : \mathbf{b}\end{bmatrix} \\ \text{H} : \mathbf{a}\end{bmatrix} \Big/ \{\langle \begin{bmatrix}\text{F} : \begin{bmatrix}\text{G} : \mathbf{b}\end{bmatrix}\end{bmatrix} \mathbf{t}\rangle\}$$

The third element in the structure is a *tail*. *Tails* record certain information from the unification history from which the TDFS was constructed. A tail consists of a set of pairs, where the first member of the pair is an atomic FS (that is, a single path or path equivalence) and the second member is a type. These pairs record default information that appeared in a TDFS that was used to build the TDFS being considered, together with the root type of that TDFS. Indeed the default information must be *strictly* default, in the sense that it was strictly more specific than the non-default information in that TDFS (so it was information that is not known for sure). The position of this root type in the type hierarchy, which we assume to be a finite bounded complete partial order (Carpenter, 1992), determines the *specificity* of the default information. So during a series of default unifications, tails will record default information from original TDFSs together with its specificity. For example, suppose a TDFS has a tail of the following form:

$$\{\ \langle\begin{bmatrix}\mathrm{F} : \begin{bmatrix}\mathrm{G} : \mathbf{b}\end{bmatrix}\end{bmatrix}, \mathbf{t}_1\rangle,\ \langle\begin{bmatrix}\mathrm{F} : \mathbf{a}\end{bmatrix}, \mathbf{t}_2\rangle,\ \langle\begin{bmatrix}\mathrm{F} : \boxed{1} \\ \mathrm{G} : \boxed{1}\end{bmatrix}, \mathbf{t}_3\rangle\}$$

This means that to produce the TDFS with this tail, we unified: a TDFS that had a root type $\mathbf{t}_1$ and the strictly defeasible path F:G:**b**; a TDFS that had root type $\mathbf{t}_2$ and strictly defeasible path F:**a**; and a TDFS that had root type $\mathbf{t}_3$ and strictly defeasible information that the F- and G-paths share the same value. These TDFSs may have contained further information; e.g., the latter TDFS may have contained the information that not only did F- and G- by default share the same values, whatever they are, but also that indefeasibly, F's value is **c** and G's value is **c**, as given in the TDFS below (ignoring the tail of this TDFS for now):

(2) $$\begin{bmatrix}\mathbf{t}_3 \\ \mathrm{F} : \mathbf{c} \\ \mathrm{G} : \mathbf{c}\end{bmatrix} \Big/ \begin{bmatrix}\mathbf{t}_3 \\ \mathrm{F} : \boxed{1} \\ \mathrm{G} : \boxed{1}\end{bmatrix}$$

The point is that tails don't record *all* the information that forms part of the unification history; it only records the strictly default information. We explain this in more detail below.

We keep track of this information in the tail, because it is the cost we must pay for being able to define default unification as a binary order independent operation. The assumption of a binary operation is problematic since by the very definition of nonmonotonicity, one cannot in general divide the premises into groups, work out the inferences from those groups, and expect the inferences to survive the whole. The only way to gain order independence, therefore, is to ensure that each time the binary operation is performed, it is influenced by the necessary pieces of information that are 'outside' the immediate context of the FSs it is unifying. Tails cover this necessary information by recording sufficient information for us to define the default unification operation so that it has all the desired properties. In Lascarides *et al* (in press), we argued in formal detail why tails were necessary. The discussion is quite lengthy and technical, and since it detracts from the main purpose of this paper, which is to produce a more powerful default unification than was presented there, we refer the reader to that paper for the relevant formal motivation.

Indeed, the tails we have defined here contain more information than those in Lascarides *et al.* (in press): specifically tails here record default reentrancies. This 'bigger' tail is the cost of defining a more powerful operation than PDU; one in which default reentrancies can be overidden by default values.

All TDFSs will have tails, even ones that are not derived from TDFSs in the initial specification, but rather are part of the initial specification itself. We show how to define tails for the initial specification shortly. And we will refer to the initial specifications of defaults as the *basic* TDFSs, and others as *derived* TDFSs.

We can make some abbreviations, so that the 'double' AVM notation can be factored across individual paths into a single AVM representation where we have *IndefeasibleValue*/*DefeasibleValue* which can be abbreviated to *IndefeasibleValue* where *IndefeasibleValue* is the same as *DefeasibleValue*, and /*DefeasibleValue* for $\top$/*DefeasibleValue*, without ambiguity ($\top$ is the most general type). We can also omit the tail when it can be determined from the default FS. That is, we omit the tail when it records exactly the strict default information in the TDFS. Thus (1) can be represented as (3).

(3) $\begin{bmatrix} \mathbf{t} & \\ \text{F} : & \begin{bmatrix} \text{G} : /\mathbf{b} \end{bmatrix} \\ \text{H} : & \mathbf{a} \end{bmatrix}$

The formal definition of TDFSs is as follows:

Definition 1 Typed Default Feature Structures
A typed default feature structure defined on a set of features Feat*, a type hierarchy* $\langle \mathsf{Type}, \sqsubseteq \rangle$ *and a set of indices* N *is a tuple* $\langle \mathcal{T}, Q, \mathcal{R}, \delta, \theta \rangle$*, where:*

- $\mathcal{T}$ *is a tail: that is, it is a set of pairs, where:*
  - *The first member of the pair is an atomic* FS*, as defined in (Carpenter, 1993); that is, a single path, or a path equivalence; and*
  - *the second member of the pair is a type;*
- Q *is a finite set of nodes,*
- $\mathcal{R} \in Q \times Q$ *(these are the root nodes of the indefeasible and defeasible* TFSs*)*
- $\theta : Q \longrightarrow \mathsf{Type}$ *is a partial typing function (this labels nodes with types).*
- $\delta : Q \times \mathsf{Feat} \longrightarrow Q$ *is a partial feature value function. (This connects nodes with arcs labelled with features).*
  Note that for notational convenience we may write a sequence of features $F_1 \ldots F_k$ *as* π*, and* $\delta(\ldots(\delta(n, F_1), F_2) \ldots F_k)$ *as* $\delta(n, \pi)$*.)*

Furthermore, the following must hold: if $\langle n_1, n_2 \rangle = \mathcal{R}$ *then*

1. n_1 *and* n_2 *aren't* δ*-descendants.*
2. *All members of* Q *are* δ*-descendants of* n_1 *or* n_2 *(but not both).*
3. *There is no node* n *or path* π *such that* $\delta(n, \pi) = n$*.*
4. $\theta(n_2) \sqsubseteq \theta(n_1)$
5. *If* $n \in Q$ *is a* δ*-descendant of* n_1*, and* $\theta(n_2) \neq \bot$*, then there is an isomorphic* $m \in Q$ *that is a* δ*-descendant of* n_2*. That is:*
   (a) *If* $\delta(n, \pi) = n$*, then* $\delta(n_2, \pi) = m$*; and*
   (b) $\theta(m) \sqsubseteq \theta(n)$

   (Conditions 1. and 2. ensure we have two separate connected directed graphs. Condition 3. ensures these two directed graphs are acyclic. And conditions 4. and 5. ensure defeasible information is at least as specific as indefeasible information).

We can extract from any TDFS the indefeasible TFS $\langle Q_I, n_1, \delta_I, \theta_I \rangle$, where $Q_I \subseteq Q$ are the δ-descendants of n_1, and δ_I and θ_I are the projections of δ and θ onto Q_I; similarly for the defeasible TFS.

3 The Definition of YADU

We now describe the operation YADU, and we will prove that it has the properties set out in section 1 in section 4.

The operation YADU, which is notated $\overset{<>}{\sqcap}$ and which operates over TDFSs, will be built up from a combination of the unification and generalisation operations $\sqcap$ and $\sqcup$ which operate on TFSs (that is, typed feature structures as defined in Carpenter (1992)). So it is necessary for us first to give a definition of these operations on TFSs. We'll then show how together with tails, they contribute to the definition of $\overset{<>}{\sqcap}$ of TDFSs.

The unification operation $\sqcap$ on TFSs is defined here as in Carpenter (1993), in terms of subsumption on typed feature structures. First a little notation: Let F be a TFS. Then $\pi \equiv_F \pi'$ means that F contains path equivalence or reentrancy between the paths π and π'; and $\mathcal{P}_F(\pi) = \sigma$ means that the value on the path π in F is σ (i.e, $\mathcal{P}_F(\pi) = \sigma$ if and only if $\theta(\delta(n, \pi)) = \sigma$, where n is the root node of F). Subsumption is then defined as follows:

Definition 2 Subsumption of typed feature structures
F subsumes F', written $F' \sqsubseteq F$, if and only if:

- $\pi \equiv_F \pi'$ *implies* $\pi \equiv_{F'} \pi'$
- $\mathcal{P}_F(\pi) = t$ *implies* $\mathcal{P}_{F'}(\pi) = t'$ *and* $t' \sqsubseteq t$

The subsumption hierarchy is a bounded complete partial order.

Unification is defined in terms of subsumption:

Definition 3 Unification
The unification $F \sqcap F'$ of two feature structures F and F' is taken to be the greatest lower bound of F and F' in the collection of feature structures ordered by subsumption.

Thus $F \sqcap F' = F''$ if and only if $F'' \sqsubseteq F$, $F'' \sqsubseteq F'$ and for every F''' such that $F''' \sqsubseteq F$ and $F''' \sqsubseteq F'$ it is also the case that $F''' \sqsubseteq F''$. Generalisation is the opposite of unification in which the lowest upper bound of two feature structures is taken. The generalisation of two feature structures is defined to be the most specific feature structure which contains only information found in both feature structures.

Definition 4 Generalisation
The generalisation $F \sqcup F'$ of two feature structures is defined to be their greatest lower bound in the subsumption ordering.

Thus $F \sqcup F' = F''$ if and only if $F \sqsubseteq F''$, $F' \sqsubseteq F''$ and for every F''' such that $F \sqsubseteq F'''$ and $F' \sqsubseteq F'''$ then $F'' \sqsubseteq F'''$.

The operation YADU on TDFSs is defined in terms of an extended version of Carpenter's (1993) credulous default unification, which in turn is defined in terms of $\sqcap$ and $\sqcup$ on TFSs. We extend it here, in that the second argument of the operation will be a set of atomic TFSs that may not be mutually compatible, instead of it being a TFS.

Carpenter's definition of credulous default unification is as follows:

Definition 5 Carpenter's Definition of Credulous Default Unification
The result of credulously adding the default information in G to the strict information in F is

Where $\mathbf{t'} \sqsubset \mathbf{t}$:

$$\begin{bmatrix}\mathbf{t}\\ \mathrm{F}:\mathbf{a}\\ \mathrm{G}:\top\end{bmatrix} \overset{<}{\sqcap} \begin{bmatrix}\mathbf{t'}\\ \mathrm{F}:\boxed{1}\,\mathbf{b}\\ \mathrm{G}:\boxed{1}\end{bmatrix} = \left\{\begin{bmatrix}\mathbf{t'}\\ \mathrm{F}:\mathbf{a}\\ \mathrm{G}:\mathbf{b}\end{bmatrix}, \begin{bmatrix}\mathbf{t'}\\ \mathrm{F}:\boxed{1}\,\mathbf{a}\\ \mathrm{G}:\boxed{1}\end{bmatrix}\right\}$$

Figure 2: Credulous Default Unification

given by:

$$F \overset{<}{\sqcap}_c G = \{F \sqcap G' : G' \sqsupseteq G \textit{ is maximal such that } F \sqcap G' \textit{ is defined }\}$$

An example of the operation is given in Figure 2. One can see that credulous default unification is asymmetric, and doesn't return a single result deterministically. So it doesn't meet the criteria we specified in section 1. However, defining $\overset{<>}{\sqcap}$ on TDFSs in terms of $\overset{<}{\sqcap}_c$ on TFSs will yield the properties we want. We extend the definition of credulous default unification, so that it takes a TFS on the left hand side and a *set* of atomic TFSs on the right hand side, where these atomic TFSs may contain mutually incompatible information:

$$F_1 \overset{<}{\sqcap}_{ca} \{G_1, \ldots, G_n\} = \{F_1 \sqcap F_2 : \; F_2 \text{ is the unification of a maximal subset of } \{G_1, \ldots, G_n\} \text{ such that } F_1 \sqcap F_2 \text{ is defined}\}$$

We will want to iterate credulous default unification in what happens below, and for this purpose, we define $\overset{<}{\sqcap}_{cs}$ as follows:

Definition 6 Credulous Default Unification $\overset{<}{\sqcap}_{cs}$ on Sets
Let $\mathcal{F}_1$ be a set *of* TFS*s $\{F_1, \ldots, F_n\}$, and $\mathcal{G}_2$ a set of atomic* TFS*s. Then*

$$\mathcal{F}_1 \overset{<}{\sqcap}_{cs} \mathcal{G}_2 = \{F_1 \overset{<}{\sqcap}_{ca} \mathcal{G}_2, \ldots, F_n \overset{<}{\sqcap}_{ca} \mathcal{G}_2\}$$

The definition of $\overset{<>}{\sqcap}$ will manipulate tails in a certain way, and the following way of partitioning tails plays an important role in defining how defaults behave under $\overset{<>}{\sqcap}$:

Definition 7 The Specificity Partition of a Tail
Let T be a tail. Then $\mu_1, \ldots \mu_m$ is a Specificity Partition of T if:

$$T = \mu_1 \cup \ldots \cup \mu_m$$

and

$$\begin{aligned} \mu_1 = \;& \{\langle \phi, t\rangle \in T : \; \forall \langle \phi', t'\rangle \in T, t' \not\sqsubset t\} \\ \mu_i = \;& \{\langle \phi, t\rangle \in T : \; \exists \langle \phi', t'\rangle \in \mu_{i-1} \textit{ where } t' \sqsubset t \textit{ and} \\ & \forall \langle \phi'', t''\rangle \in T \setminus (\mu_1 \cup \ldots \cup \mu_{i-1}), t'' \not\sqsubset t\} \quad 2 \leq i \leq m \end{aligned}$$

Intuitively, μ_1 is the set of pairs in T with the most specific types, μ_2 is the set of pairs in T with the next most specific types, and so on. Partitioning tails in order of specificity enables us to define $\overset{<>}{\sqcap}$ so that default information on more specific types overrides that on more general types.

We will also need to talk about the first and second members of the pairs in the tail:

Definition 8 Projection of Tails
Let T be a tail; i.e., a member of atomic FS*s $\times$* Type. *Then:*

$$\begin{aligned} \wp_{fs}(T) = \;& \{\phi : \langle \phi, t\rangle \in T\} \textit{ and} \\ \wp_t(T) = \;& \{t : \langle \phi, t\rangle \in T\} \end{aligned}$$

Having set up this groundwork, we're now in a position to informally define $\overset{<>}{\sqcap}$ on TDFSs in terms of $\sqcup$ and $\overset{<}{\sqcap}_{cs}$ on TFSs. Let $F =_{def} I/D/T$ be a TDFS, and so I is the indefeasible TFS, D the defeasible TFS and T the tail. Let F_1 and F_2 be two TDFSs, and let $F_{12} =_{def} F_1 \overset{<>}{\sqcap} F_2$. Furthermore, assume $F_{12} =_{def} I_{12}/D_{12}/T^{12}$. then I_{12}, D_{12} and T^{12} are calculated as follows:

1. The Indefeasible Part:
   $I_{12} = I_1 \sqcap I_2$
   That is, the indefeasible TFS is the unification of the indefeasible parts of the arguments.

2. The Tail T^{12}:
   $T^{12} =_{def} (T^1 \cup T^2) \setminus Bot^{12}$
   where Bot^{12} are all the elements of $T^1 \cup T^2$ where the atomic FS is incompatible with I_{12}. That is:
   $$Bot^{12} = \{\langle F, \mathbf{t}\rangle : \langle F, \mathbf{t}\rangle \in T^1 \cup T^2 \text{ and } F \sqcap I_{12} = \bot\}$$
   Note that if $I/D/T$ is a basic TDFS, then its tail T is defined as follows:
   $$\begin{aligned} T = \{\langle \phi, t\rangle : \quad & \text{where } t \text{ is the root type on } D \text{ and} \\ & \phi \text{ is an atomic FS such that } \phi \sqsupseteq D \text{ but } \phi \not\sqsupseteq I\} \end{aligned}$$

3. The Defeasible Part: D_{12} is defined in terms of I_{12} and T^{12}. We first divide T^{12} into a specificity partition $\langle \mu_1, \ldots, \mu_n\rangle$. Then
   $$D_{12} = \sqcup((I_{12} \overset{<}{\sqcap}_{cs} \wp_{fs}(\mu_1)) \overset{<}{\sqcap}_{cs} \ldots \overset{<}{\sqcap}_{cs} \wp_{fs}(\mu_n))$$
   That is, D_{12} is the generalisation of the 'credulous default unification' of: I_{12}; the atomic FSs in μ_1; $\ldots$; the atomic FSs in μ_m, in that order.

3.1 Examples

We will prove that the operation has the properties specified in section 1 in the next section. But first, we will illustrate how the operation works by means of specific examples.

3.1.1 Specificity

The following example demonstrates that default reentrancies are overridden by conflicting default information of a more specific type.

Where $\mathbf{t} \sqsubset \mathbf{t'}$:

$$\begin{bmatrix} \mathbf{t} & \\ \text{F} : & /\mathbf{a} \\ \text{G} : & /\mathbf{b} \end{bmatrix} \overset{<>}{\sqcap} \begin{bmatrix} \mathbf{t'} & \\ \text{F} : & /\boxed{1} \\ \text{G} : & /\boxed{1} \end{bmatrix}$$

- $I_{12} = \begin{bmatrix} \mathbf{t} & \\ \text{F} : & \top \\ \text{G} : & \top \end{bmatrix} \sqcap \begin{bmatrix} \mathbf{t'} & \\ \text{F} : & \top \\ \text{G} : & \top \end{bmatrix} = \begin{bmatrix} \mathbf{t} & \\ \text{F} : & \top \\ \text{G} : & \top \end{bmatrix}$

- $$\begin{aligned} T^{12} &= (T^1 \cup T^2) \setminus Bot^{12} \\ T^1 &= \{\langle [\text{F} : \mathbf{a}], \mathbf{t}\rangle, \langle [\text{G} : \mathbf{b}], \mathbf{t}\rangle\} \\ T^2 &= \{\langle \begin{bmatrix} \text{F} : \boxed{1} \\ \text{G} : \boxed{1} \end{bmatrix}, \mathbf{t'}\rangle\} \\ Bot^{12} &= \emptyset \\ \text{So} & \\ T^{12} &= (T^1 \cup T^2) \setminus Bot^{12} \\ &= \{\langle [\text{F} : \mathbf{a}], \mathbf{t}\rangle, \langle [\text{G} : \mathbf{b}], \mathbf{t}\rangle, \langle \begin{bmatrix} \text{F} : \boxed{1} \\ \text{G} : \boxed{1} \end{bmatrix}, \mathbf{t'}\rangle\} \end{aligned}$$

Therefore the specificity partition of T^{12} is $\langle \mu_1, \mu_2 \rangle$, where:

$$\begin{aligned} \mu_1 = &\ \{\langle \begin{bmatrix} \mathrm{F} : \mathbf{a} \end{bmatrix}, \mathbf{t} \rangle, \langle \begin{bmatrix} \mathrm{G} : \mathbf{b} \end{bmatrix}, \mathbf{t} \rangle\} \\ \mu_2 = &\ \{\langle \begin{bmatrix} \mathrm{F} : \boxed{1} \\ \mathrm{G} : \boxed{1} \end{bmatrix}, \mathbf{t'} \rangle\} \end{aligned}$$

- $$\begin{aligned} D_{12} = &\ \sqcup((I_{12} \stackrel{<}{\sqcap}_{cs} \wp_{fs}(\mu_1)) \stackrel{<}{\sqcap}_{cs} \wp_{fs}(\mu_2)) \\ = &\ \sqcup \left(\left(\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \top \\ \mathrm{G} : \top \end{bmatrix} \stackrel{<}{\sqcap}_{cs} \left\{ \begin{bmatrix} \mathrm{F} : \mathbf{a} \end{bmatrix}, \begin{bmatrix} \mathrm{G} : \mathbf{b} \end{bmatrix} \right\} \right) \stackrel{<}{\sqcap}_{cs} \begin{bmatrix} \mathrm{F} : \boxed{1} \\ \mathrm{G} : \boxed{1} \end{bmatrix} \right) \\ = &\ \sqcup \left(\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \mathbf{a} \\ \mathrm{G} : \mathbf{b} \end{bmatrix} \stackrel{<}{\sqcap}_{cs} \begin{bmatrix} \mathrm{F} : \boxed{1} \\ \mathrm{G} : \boxed{1} \end{bmatrix} \right) \\ = &\ \begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \mathbf{a} \\ \mathrm{G} : \mathbf{b} \end{bmatrix} \end{aligned}$$

Hence

$$\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : /\mathbf{a} \\ \mathrm{G} : /\mathbf{b} \end{bmatrix} \stackrel{<>}{\sqcap} \begin{bmatrix} \mathbf{t'} \\ \mathrm{F} : /\boxed{1} \\ \mathrm{G} : /\boxed{1} \end{bmatrix} = \begin{bmatrix} \mathbf{t} \\ \mathrm{F} : /\mathbf{a} \\ \mathrm{G} : /\mathbf{b} \end{bmatrix} \Big/ \ \{\langle \begin{bmatrix} \mathrm{F} : \mathbf{a} \end{bmatrix}, t \rangle, \langle \begin{bmatrix} \mathrm{G} : \mathbf{b} \end{bmatrix}, t \rangle, \langle \begin{bmatrix} \mathrm{F} : \boxed{1} \\ \mathrm{G} : \boxed{1} \end{bmatrix}, t' \rangle\}$$

3.1.2 The Nixon Diamond

The credulous unification operation $\stackrel{<}{\sqcap}_{cs}$ doesn't generally return a deterministic result. However, $\stackrel{<>}{\sqcap}$ does, even though it's defined in terms of $\stackrel{<}{\sqcap}_{cs}$. This is because $\stackrel{<>}{\sqcap}$ generalises over the credulous options. The following example illustrates this.

Where $\mathbf{t}$ and $\mathbf{t'}$ are in no specificity ordering, and

$$\begin{aligned} \mathbf{t} \sqcap \mathbf{t'} = &\ \mathbf{t''} \\ \mathbf{a} \sqcap \mathbf{b} = &\ \bot \\ \mathbf{a} \sqcup \mathbf{b} = &\ \mathbf{c} \end{aligned}$$

$$\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : /\mathbf{a} \\ \mathrm{G} : /\mathbf{b} \end{bmatrix} \stackrel{<>}{\sqcap} \begin{bmatrix} \mathbf{t'} \\ \mathrm{F} : /\boxed{1} \\ \mathrm{G} : /\boxed{1} \end{bmatrix}$$

- $$\begin{aligned} I_{12} = &\ \begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \top \\ \mathrm{G} : \top \end{bmatrix} \sqcap \begin{bmatrix} \mathbf{t'} \\ \mathrm{F} : \top \\ \mathrm{G} : \top \end{bmatrix} \\ = &\ \begin{bmatrix} \mathbf{t''} \\ \mathrm{F} : \top \\ \mathrm{G} : \top \end{bmatrix} \end{aligned}$$

- $$\begin{aligned} T^{12} &= (T^1 \cup T^2) \setminus Bot^{12} \\ T^1 &= \{\langle \begin{bmatrix} \mathrm{F} : \mathbf{a} \end{bmatrix}, \mathbf{t} \rangle, \langle \begin{bmatrix} \mathrm{G} : \mathbf{b} \end{bmatrix}, \mathbf{t} \rangle\} \\ T^2 &= \{\langle \begin{bmatrix} \mathrm{F} : \boxed{1} \\ \mathrm{G} : \boxed{1} \end{bmatrix}, \mathbf{t'} \rangle\} \\ Bot^{12} &= \emptyset \\ \text{So} & \\ T^{12} &= (T^1 \cup T^2) \setminus Bot^{12} \\ &= \{\langle \begin{bmatrix} \mathrm{F} : \mathbf{a} \end{bmatrix}, \mathbf{t} \rangle, \langle \begin{bmatrix} \mathrm{G} : \mathbf{b} \end{bmatrix}, \mathbf{t} \rangle, \langle \begin{bmatrix} \mathrm{F} : \boxed{1} \\ \mathrm{G} : \boxed{1} \end{bmatrix}, \mathbf{t'} \rangle\} \end{aligned}$$

  Therefore the specificity partition of T^{12} is T^{12} itself (since both $\mathbf{t}$ and $\mathbf{t'}$ are the most specific types in T^{12}).

- $D_{12} = \sqcup((I_{12} \stackrel{<}{\sqcap}_{cs} \wp_{fs}(T^{12})))$
$= \sqcup\left(\left(\begin{bmatrix}\textbf{t''}\\ \text{F} : \top\\ \text{G} : \top\end{bmatrix} \stackrel{<}{\sqcap}_{cs} \left\{ \begin{bmatrix}\text{F} : \textbf{a}\end{bmatrix}, \begin{bmatrix}\text{G} : \textbf{b}\end{bmatrix}, \begin{bmatrix}\text{F} : \boxed{1}\\ \text{G} : \boxed{1}\end{bmatrix} \right\}\right)\right)$
$= \sqcup\left(\left\{ \begin{bmatrix}\textbf{t''}\\ \text{F} : \textbf{a}\\ \text{G} : \textbf{b}\end{bmatrix}, \begin{bmatrix}\textbf{t''}\\ \text{F} : \boxed{1}\textbf{a}\\ \text{G} : \boxed{1}\end{bmatrix}, \begin{bmatrix}\textbf{t''}\\ \text{F} : \boxed{1}\textbf{b}\\ \text{G} : \boxed{1}\end{bmatrix} \right\}\right)$
$= \begin{bmatrix}\textbf{t''}\\ \text{F} : \textbf{c}\\ \text{G} : \textbf{c}\end{bmatrix}$

Hence

$$\begin{bmatrix}\textbf{t}\\ \text{F} : /\textbf{a}\\ \text{G} : /\textbf{b}\end{bmatrix} \stackrel{<>}{\sqcap} \begin{bmatrix}\textbf{t'}\\ \text{F} : /\boxed{1}\\ \text{G} : /\boxed{1}\end{bmatrix} = \begin{bmatrix}\textbf{t}\\ \text{F} : /\textbf{c}\\ \text{G} : /\textbf{c}\end{bmatrix} \Big/ \{\langle \begin{bmatrix}\text{F} : \textbf{a}\end{bmatrix}, \textbf{t}\rangle, \langle \begin{bmatrix}\text{G} : \textbf{b}\end{bmatrix}, \textbf{t}\rangle, \langle \begin{bmatrix}\text{F} : \boxed{1}\\ \text{G} : \boxed{1}\end{bmatrix}, \textbf{t'}\rangle\}$$

3.1.3 Well-Typed Unification

Unlike the order independent typed default unification operation PDU defined in Lascarides *et al.* (in press), $\stackrel{<>}{\sqcap}$ never produces TDFSs where there is an atomic type/compound clash. PDU was problematic in that the following held:

Where $\textbf{t} \sqsubset \textbf{t'}$ and $\textbf{a}$ is an atomic type:

$$\begin{bmatrix}\textbf{t}\\ \text{F} : /\textbf{a}\end{bmatrix} \text{PDU} \begin{bmatrix}\textbf{t'}\\ \text{F} : /\begin{bmatrix}\textbf{t1}\\ \text{G} : \textbf{b}\end{bmatrix}\end{bmatrix} = \begin{bmatrix}\textbf{t}\\ \text{F} : /\begin{bmatrix}\textbf{a}\sqcap\textbf{t1}\\ \text{G} : \textbf{b}\end{bmatrix}\end{bmatrix}$$

So combining PDU with operations that ensure the result is always well-typed was not investigated. This problem with PDU stemmed from the fact that the defeasible results of the operation for each path were calculated independently. This made the operation polynomial, but produced odd results in cases like those above. In contrast, $\stackrel{<>}{\sqcap}$ here is not calculated on a path-by-path basis. We'll demonstrate here that as a consequence, we don't get atomic/compound clashes. But the cost of this is exponential complexity, since $\stackrel{<}{\sqcap}_c$, on which $\stackrel{<>}{\sqcap}$ is based, is exponential.

So, let's calculate the following:

Where $\textbf{t} \sqsubset \textbf{t'}$

$$\begin{bmatrix}\textbf{t}\\ \text{F} : /\textbf{a}\end{bmatrix} \stackrel{<>}{\sqcap} \begin{bmatrix}\textbf{t'}\\ \text{F} : /\begin{bmatrix}\textbf{t1}\\ \text{G} : \textbf{b}\end{bmatrix}\end{bmatrix}$$

- $I_{12} = I_1 \sqcap I_2 = \begin{bmatrix}\textbf{t}\\ \text{F} : \top\end{bmatrix}$
- $T^{12} = (T^1 \cup T^2) \setminus Bot^{12}$
$T^1 = \{\langle \begin{bmatrix}\text{F} : \textbf{a}\end{bmatrix}, \textbf{t}\rangle\}$
$T^2 = \{\langle \begin{bmatrix}\text{F} : \textbf{t1}\end{bmatrix}, \textbf{t'}\rangle, \langle \begin{bmatrix}\text{F} : \begin{bmatrix}\text{G} : \textbf{b}\end{bmatrix}\end{bmatrix}, \textbf{t'}\rangle\}$
$Bot^{12} = \emptyset$
So
$T^{12} = (T^1 \cup T^2) \setminus Bot^{12}$
$= \{\langle \begin{bmatrix}\text{F} : \textbf{a}\end{bmatrix}, \textbf{t}\rangle, \langle \begin{bmatrix}\text{F} : \textbf{t1}\end{bmatrix}, \textbf{t'}\rangle, \langle \begin{bmatrix}\text{F} : \begin{bmatrix}\text{G} : \textbf{b}\end{bmatrix}\end{bmatrix}, \textbf{t'}\rangle\}$

Therefore the specificity partition of T^{12} is $\langle \mu_1, \mu_2\rangle$, where:

$$\mu_1 = \{\langle \begin{bmatrix}\text{F} : \textbf{a}\end{bmatrix}, \textbf{t}\rangle\}$$
$$\mu_2 = \{\langle \begin{bmatrix}\text{F} : \textbf{t1}\end{bmatrix}, \textbf{t'}\rangle, \langle \begin{bmatrix}\text{F} : \begin{bmatrix}\text{G} : \textbf{b}\end{bmatrix}\end{bmatrix}, \textbf{t'}\rangle\}$$

- $$\begin{aligned} D_{12} = \quad & \sqcup(((I_{12} \overset{<}{\sqcap}_{cs} \wp_{fs}(\mu_1)) \overset{<}{\sqcap}_{cs} \wp_{fs}(\mu_2)) \\ = \quad & \sqcup\left(\left(\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \top \end{bmatrix} \overset{<}{\sqcap}_{cs} \left\{ \begin{bmatrix} \mathrm{F} : \mathbf{a} \end{bmatrix} \right\} \right) \overset{<}{\sqcap}_{cs} \left\{ \begin{bmatrix} \mathrm{F} : \mathbf{t1} \end{bmatrix}, \begin{bmatrix} \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{b} \end{bmatrix} \end{bmatrix} \right\} \right) \\ = \quad & \sqcup\left(\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \mathbf{a} \end{bmatrix} \overset{<}{\sqcap}_{cs} \left\{ \begin{bmatrix} \mathrm{F} : \mathbf{t1} \end{bmatrix}, \begin{bmatrix} \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{b} \end{bmatrix} \end{bmatrix} \right\} \right) \\ = \quad & \begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \mathbf{a} \sqcap \mathbf{t1} \end{bmatrix} \end{aligned}$$

Hence

$$\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : /\mathbf{a} \end{bmatrix} \overset{<>}{\sqcap} \begin{bmatrix} \mathbf{t'} \\ \mathrm{F} : / \begin{bmatrix} \mathbf{t1} \\ \mathrm{G} : \mathbf{b} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \mathbf{t} \\ \mathrm{F} : /\mathbf{a} \sqcap \mathbf{t1} \end{bmatrix} \Big/ \ \{\langle \begin{bmatrix} \mathrm{F} : \mathbf{a} \end{bmatrix}, \mathbf{t} \rangle, \langle \begin{bmatrix} \mathrm{F} : \mathbf{t1} \end{bmatrix}, \mathbf{t'} \rangle, \langle \begin{bmatrix} \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{b} \end{bmatrix} \end{bmatrix}, \mathbf{t'} \rangle\}$$

4 The Properties of $\overset{<>}{\sqcap}$

We now demonstrate that $\overset{<>}{\sqcap}$ has the desirable properties we specified in section 1. In what follows, we use the following notational convention: we refer to $F_1 \overset{<>}{\sqcap} F_2$ as F_{12}, and $F_i =_{def} I_i/D_i/T^i$.

Lemma 1 *Tails are commutative.*

Proof

This follows trivially, from the commutativity of Bot^{12} and set union.

Lemma 2 *Tails are associative. That is:* $T^{(12)3} = T^{1(23)}$

Proof

$$\begin{aligned} T^{(12)3} = \quad & (T^{12} \cup T^3) \setminus Bot^{(12)3} \\ Bot^{(12)3} = \quad & \{\langle F, t \rangle \in T^{12} \cup T^3 \text{ such that } I_{(12)3} \sqcap F = \bot\} \\ = \quad & \{\langle F, t \rangle \in T^{12} \text{ such that } I_{(12)3} \sqcap F = \bot\} \cup \\ & \{\langle F, t \rangle \in T^3 \text{ such that } I_{(12)3} \sqcap F = \bot\} \end{aligned}$$

Let $\langle F, t \rangle \in T^{(12)3}$. Then:

(a) $\langle F, t \rangle \in T^{12} \setminus Bot^{(12)3}$; or

(b) $\langle F, t \rangle \in T^3 \setminus Bot^{(12)3}$

Suppose (b). Then $\langle F, t \rangle \in T^3$ and $I_{(12)3} \sqcap F \neq \bot$.

Therefore, $\langle F, t \rangle \in T^2 \cup T^3$. Furthermore, by the definition of typed unification, $I_{(12)3} = I_{1(23)} \sqsubseteq I_{23}$. Therefore, since $I_{1(23)} \sqcap F \neq \bot$, $I_{23} \sqcap F \neq \bot$. So $\langle F, t \rangle \in T^{23}$. Furthermore, $I_{1(23)} \sqcap F = I_{(12)3} \sqcap F \neq \bot$. So $\langle F, t \rangle \notin Bot^{1(23)}$, and therefore $\langle F, t \rangle \in T^{1(23)}$.

Now suppose (a) holds. Then $\langle F, t \rangle \in ((T^1 \cup T^2) \setminus Bot^{12}) \setminus Bot^{(12)3}$. But $Bot^{12} \subseteq Bot^{(12)3}$. So either:

(i) $\langle F, t \rangle \in T^1 \setminus Bot^{(12)3}$; or

(ii) $\langle F, t \rangle \in T^2 \setminus Bot^{(12)3}$

Suppose (i) holds. Then $\langle F, t\rangle \in T^1$ and $I_{(12)3} \sqcap F \neq \bot$. So $I_{1(23)} \sqcap F \neq \bot$ by the order independence of typed unification. So $\langle F, t\rangle \in T^1 \setminus Bot^{1(23)} \subseteq T^{1(23)}$.

Suppose (ii) holds. Then $\langle F, t\rangle \in T^2$ and $I_{(12)3} \sqcap F \neq \bot$. So $I_{1(23)} \sqcap F \neq \bot$, and therefore $I_{23} \sqcap F \neq \bot$. So $\langle F, t\rangle \notin Bot^{23}$, and so $\langle F, t\rangle \in T^{23}$. Furthermore, since $I_{1(23)} \sqcap F \neq \bot$, $\langle F, t\rangle \notin Bot^{1(23)}$. And so $\langle F, t\rangle \in T^{1(23)}$.

So $T^{(12)3} \subseteq T^{1(23)}$.
By symmetry, $T^{1(23)} \subseteq T^{(12)3}$.

Therefore $T^{1(23)} = T^{1(23)}$. $\square$

Lemma 3 *For any* TDFS $F = I/D/T$, $D = \bot$ *or*

$$D = \sqcup((I \overset{<}{\sqcap}_{cs} \wp_{fs}(\mu_1)) \ldots \wp_{fs}(\mu_m))$$

Where $\langle \mu_1, \ldots, \mu_m\rangle$ *is a specificity partition of* T.

Proof: If $D = \bot$ we're done. So we assume $D \neq \bot$. Then there are two cases:

(a) The TDFS F is a basic TDFS; or

(b) The TDFS F is a derived TDFS.

Case (a): By the definition of basic tails and TDFSs:

$$D = I \sqcap \wp_{fs}(T)$$

Furthermore, by the definition of basic tails, we know:

(i) T is its own specificity partition;

(ii) since $D \neq \bot$, $\sqcap \wp_{fs}(T) \neq \bot$; and

(iii) $\forall F \in \wp_{fs}(T)$, $I \sqcap F \neq \bot$.

So by the definition of $\sqcup$ and $\overset{<}{\sqcap}_{cs}$:

$$D = I \sqcap \wp_{fs}(T) = \sqcup(I \overset{<}{\sqcap}_{cs} \wp_{fs}(T))$$

Case (b): The result follows immediately by the definition of YADU, given in section 3. $\square$.

Lemma 4 $\overset{<>}{\sqcap}$ *is a function. That is* $\text{TDFS}_1 \overset{<>}{\sqcap} \text{TDFS}_2$ *is a unique* TDFS.

Proof : The indefeasible TFS of $\text{TDFS}_1 \overset{<>}{\sqcap} \text{TDFS}_2$ is unique because $\sqcap$ is deterministic. The tail is unique because set union and $\sqcap$ are deterministic. The specificity partition of a tail is unique because the type hierarchy is a complete partial order. And so the defeasible part of $\text{TDFS}_1 \overset{<>}{\sqcap} \text{TDFS}_2$ is unique because the specificity partion of a tail is unique and $\sqcup$ is deterministic. $\square$

Lemma 5 $\overset{<>}{\sqcap}$ *is commutative.*

Proof: Let us consider the $\overset{<>}{\sqcap}$ of the TDFSs F_1 and F_2. The indefeasible part of $F_1 \overset{<>}{\sqcap} F_2$ is the same as the indefeasible part of $F_2 \overset{<>}{\sqcap} F_1$ because $\sqcap$ is commutative. The tails $T^{12} = T^{21}$ by lemma 1. And the defeasible part is defined in terms of the indefeasible part and the tail. But $I_{12} = I_{21}$ and $T^{12} = T^{21}$ by lemma 1. So $D_{12} = D_{21}$. So $\overset{<>}{\sqcap}$ is commutative. □

Lemma 6 $\overset{<>}{\sqcap}$ *is associative.*

Proof: One needs to prove:

1. $I_{(12)3} = I_{1(23)}$
2. $D_{(12)3} = D_{1(23)}$
3. $T^{(12)3} = T^{(12)3}$

Case 1 follows immediately from the associativity of $\sqcap$. Case 3 holds by lemma 2. Case 2 holds by lemma 3: $D_{(12)3}$ is defined in terms of $I_{(12)3}$ and $T^{(12)3}$ which by cases 1 and 3 are the same as $I_{1(23)}$ and $T^{1(23)}$, and so $D_{(12)3} = D_{1(23)}$. So $\overset{<>}{\sqcap}$ is associative. □

Theorem 1 $\overset{<>}{\sqcap}$ *is an order independent binary function.*

Proof: Follows immediately from lemmas 4, 5 and 6.

The following theorem demonstrates that $\overset{<>}{\sqcap}$ meets criterion 2 in section 1: $\overset{<>}{\sqcap}$ reduces to $\sqcap$ when there's no conflict.

Theorem 2 The Typed Unification Property *Let F_1, F_2 be* TDFS*s. And suppose $I_{1\wedge 2} \sqcap \wp_{fs}(T^1) \sqcap \wp_{fs}(T^2) \neq \bot$. Then $D_{1\wedge 2} = D_1 \sqcap D_2$.*

Before we prove this, note that theorem 2 does *not* specify that $\overset{<>}{\sqcap}$ reduces to $\sqcap$ when there's no conflict in the defeasible TFSs. This is because the condition of the theorem is that $I_{12} \sqcap \wp_{fs}(T^1) \sqcap \wp_{fs}(T^2) \neq \bot$, and this is a stronger condition than $D_1 \sqcap D_2 \neq \bot$. The result deson't hold for the latter condition alone, as the following example demonstrates:

Where

$$\begin{array}{l} \mathbf{t1} \sqsubset \mathbf{t2} \sqsubset \mathbf{t3} \\ \mathbf{a} \sqcap \mathbf{c} = \bot \\ \mathbf{b} \sqcap \mathbf{c} = \bot \\ \mathbf{a} \sqcup \mathbf{b} = \top \end{array}$$

$$\begin{bmatrix} \mathbf{t1} \\ \mathrm{F} : /\boxed{1} \\ \mathrm{G} : /\boxed{1} \end{bmatrix} \Big/ \ \{\langle \begin{bmatrix} \mathrm{F} : \boxed{1} \\ \mathrm{G} : \boxed{1} \end{bmatrix}, \mathbf{t1}\rangle, \langle [\ \mathrm{F} : \mathbf{a}\ ], \mathbf{t2}\rangle, \langle [\ \mathrm{G} : \mathbf{b}\ ], \mathbf{t2}\rangle\} \ \overset{<>}{\sqcap} \ \begin{bmatrix} \mathbf{t3} \\ \mathrm{F} : /\mathbf{c} \\ \mathrm{G} : \top \end{bmatrix} \Big/ \ \{\langle [\ \mathrm{F} : \mathbf{c}\ ], \mathbf{t3}\rangle\}$$

We just show how to calculate the defeasible TFS:

$$
\begin{aligned}
D_{12} = \quad & \sqcup\Bigg(\Bigg(\Bigg(\begin{bmatrix} \textbf{t'} \\ \text{F} : \top \\ \text{G} : \top \end{bmatrix} \overset{<}{\sqcap}_{cs} \left\{ \begin{bmatrix} \text{F} : \boxed{1} \\ \text{G} : \boxed{1} \end{bmatrix} \right\} \Bigg) \overset{<}{\sqcap}_{cs} \left\{ \begin{bmatrix} \text{F} : \textbf{a} \end{bmatrix}, \begin{bmatrix} \text{G} : \textbf{b} \end{bmatrix} \right\} \Bigg) \overset{<}{\sqcap}_{cs} \left\{ \begin{bmatrix} \text{F} : \textbf{c} \end{bmatrix} \right\} \Bigg) \\
= \quad & \sqcup\Bigg(\Bigg(\begin{bmatrix} \textbf{t'} \\ \text{F} : \boxed{1} \\ \text{G} : \boxed{1} \end{bmatrix} \overset{<}{\sqcap}_{cs} \left\{ \begin{bmatrix} \text{F} : \textbf{a} \end{bmatrix}, \begin{bmatrix} \text{G} : \textbf{b} \end{bmatrix} \right\} \Bigg) \overset{<}{\sqcap}_{cs} \left\{ \begin{bmatrix} \text{F} : \textbf{c} \end{bmatrix} \right\} \Bigg) \\
= \quad & \sqcup\Bigg(\left\{ \begin{bmatrix} \textbf{t'} \\ \text{F} : \boxed{1}\,\textbf{a} \\ \text{G} : \boxed{1} \end{bmatrix}, \begin{bmatrix} \textbf{t'} \\ \text{F} : \boxed{1}\,\textbf{b} \\ \text{G} : \boxed{1} \end{bmatrix} \right\} \overset{<}{\sqcap}_{cs} \left\{ \begin{bmatrix} \text{F} : \textbf{c} \end{bmatrix} \right\} \Bigg) \\
= \quad & \begin{bmatrix} \textbf{t'} \\ \text{F} : \boxed{1}\,\textbf{a} \\ \text{G} : \boxed{1} \end{bmatrix} \sqcup \begin{bmatrix} \textbf{t'} \\ \text{F} : \boxed{1}\,\textbf{b} \\ \text{G} : \boxed{1} \end{bmatrix} \\
= \quad & \begin{bmatrix} \textbf{t'} \\ \text{F} : \boxed{1} \\ \text{G} : \boxed{1} \end{bmatrix}
\end{aligned}
$$

Thus $D_{12} \neq D_1 \sqcap D_2$ in this case, even though

$$D_1 \sqcap D_2 = \begin{bmatrix} \textbf{t'} \\ \text{F} : \boxed{1}\,\textbf{c} \\ \text{G} : \boxed{1} \end{bmatrix} \neq \bot$$

Proof of Theorem 2:

$T^{12} = T^1 \cup T^2$, because $I_{12} \sqcap \wp_1(T^1) \neq \bot$, and $I_{12} \sqcap \wp_1(T^2) \neq \bot$.

Let $\mu_1 \ldots \mu_n$ be a Specificity Partition Tail of T^{12}. Then by lemma 3:

$$D_{12} = \sqcup(I_{12} \overset{<}{\sqcap}_{cs} \wp_{fs}(\mu_1) \overset{<}{\sqcap}_{cs} \ldots \overset{<}{\sqcap}_{cs} \wp_{fs}(\mu_n))$$

But by assumption $I_{12} \sqcap \wp_{fs}(\mu_1) \neq \bot$. So by the definition of $\overset{<}{\sqcap}_{cs}$:

$$I_{12} \overset{<}{\sqcap}_{cs} \wp_{fs}(\mu_1) = I_{12} \sqcap \wp_{fs}(\mu_1) \neq \bot$$

Similarly $(I_{12} \sqcap \wp_{fs}(\mu_1)) \sqcap \wp_{fs}(\mu_2) \neq \bot$. So by the definition of $\overset{<}{\sqcap}_{cs}$:

$$((I_{12} \overset{<}{\sqcap}_{cs} \wp_{fs}(\mu_1)) \overset{<}{\sqcap}_{cs} \wp_{fs}(\mu_2) = I_{12} \sqcap \wp_{fs}(\mu_1) \sqcap \wp_{fs}(\mu_2)$$

By similar arguments for $\mu_3, \ldots, \mu_n$:

$$
\begin{aligned}
D_{12} = \quad & \sqcup((I_{12} \overset{<}{\sqcap}_{cs} \wp_{fs}(\mu_1)) \ldots \overset{<}{\sqcap}_{cs} \wp_{fs}(\mu_n)) \\
= \quad & \sqcup(I_{12} \sqcap \wp_{fs}(\mu_1) \sqcap \ldots \sqcap \wp_{fs}(\mu_n)) \\
= \quad & I_{12} \sqcap \wp_{fs}(T^{12}) \\
= \quad & I_1 \sqcap \wp_{fs}(T^1) \sqcap I_2 \sqcap \wp_{fs}(T^2)
\end{aligned}
$$

By a similar argument to that above:

$$
\begin{aligned}
D_1 = \quad & I_1 \sqcap \wp_1(T^1) \\
D_2 = \quad & I_2 \sqcap \wp_1(T^2)
\end{aligned}
$$

So

$$D_{12} = D_1 \sqcap D_2$$

as required. □

Because TDFSs contain more information than the information in the FS components alone, the notion of 'conflict' is more complex than in Carpenter (1992, 1993) and Bouma (1992). Their definitions of default unification correspond to the situations where we are unifying structures in the initial KB—that is, basic TDFSs. and we can prove that for these cases the typed unification property is one where $\overset{<>}{\sqcap}$ reduces to $\sqcap$ when there is no conflict among the FS components:

Corollary 1 The Typed Unification Property for Basic TDFSs
If TDFS_1 *and* TDFS_2 *are basic* TDFSs*, and* $D_1 \sqcap D_2 \neq \bot$*, then*

$$D_{12} = D_1 \sqcap D_2$$

Proof: Since TDFS_1 and TDFS_2 are basic: $D_i = I_i \sqcap \wp_{fs}(T^i)$ for $i = 1, 2$. So, since $D_1 \sqcap D_2 \neq \bot$:

$$\begin{aligned} I_1 \sqcap \wp_{fs}(T^1) \sqcap I_2 \sqcap \wp_{fs}(T^2) = &\quad I_{12} \sqcap \wp_{fs}(T^1) \sqcap \wp_{fs}(T^2) \\ \neq &\quad \bot \end{aligned}$$

So by theorem 2:

$$D_{12} = D_1 \sqcap D_2$$

□

5 Conclusion

We have defined a version of default unification on typed feature structures which is both order independent and sufficiently expressive to allow overriding of any default information by more specific conflicting defaults. This has numerous potential applications in computational linguistics, both in improving the declarativity of existing uses of default inheritance within the lexicon, and allowing defaults to persist beyond the lexicon (Lascarides *et al*, in press). The worst-case complexity properties of YADU are unpromising since it involves multiple applications of Carpenter's credulous default unification operation, which is apparently NP-complete. Many applications involve rather few default statements, however, and if these tend not to interact, the operation won't be intractable in practice. In future work, we will demonstrate the utility of YADU in several linguistic applications.

References

Asher, N. and Morreau, M. (1991) 'Common Sense Entailment: A Modal Theory of Nonmonotonic Reasoning', *Proceedings of the Proceedings of the 12th International Joint Conference on Artificial Intelligence,* Sydney Australia, pp. .

Blackburn, P. (1992) 'Modal Logic and Attribute Value Structures' in M. de Rijke (eds.), *Diamonds and Defaults,* Studies in Logic, Language and Information. Kluwer, Dordrecht, Holland (available as University of Amsterdam, ITLI, LP-92-02).

Boguraev, B. and J. Pustejovsky (1990) 'Lexical ambiguity and the role of knowledge representation in lexicon design', *Proceedings of the 13th Int. Conf. on Comp. Ling. (COLING-90),* Helsinki, pp. 36–42.

Bouma, G. (1992) 'Feature structures and nonmonotonicity', *Computational Linguistics, vol.18.2,* 183–204.

Briscoe, E.J., A. Copestake and B.K. Boguraev (1990) 'Enjoy the paper: Lexical semantics via lexicology', *Proceedings of the 13th Int. Conf. on Comp. Ling. (COLING-90),* Helsinki, pp. 42–47.

Carpenter, R. (1992) *The logic of typed feature structures,* (Tracts in Theoretical Computer Science), Cambridge University Press, Cambridge, England.

Carpenter, R. (1993) 'Skeptical and credulous default unification with application to templates and inheritance' in E. J. Briscoe, A. Copestake and V. de Paiva (eds.), *Inheritance, Defaults and the Lexicon,* Cambridge University Press, Cambridge, England, pp. 13–37.

Copestake, A. (1992) *The Representation of Lexical Semantic Information,* Doctoral dissertation, University of Sussex, Cognitive Science Research Paper CSRP 280.

Copestake, A. (1993) 'Defaults in lexical representation' in E. J. Briscoe, A. Copestake and V. de Paiva (eds.), *Inheritance, Defaults and the Lexicon,* Cambridge University Press, Cambridge, England, pp. 223–245.

Daelemans, W. (1987) 'A tool for the automatic creation, extension and updating of lexical knowledge bases', *Proceedings of the 3rd Conference of the European Chapter of the Association for Computational Linguistics (EACL-87),* Copenhagen, pp. 70–74.

Daelemans, W., K. de Smedt and G. Gazdar (1992) 'Inheritance in natural language processing', *Computational Linguistics, vol.18.2,* 205–218.

Evans, R. and G. Gazdar (1989a) 'Inference in DATR', *Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics (EACL-1989),* Manchester, England, pp. 66–71.

Evans, R. and G. Gazdar (1989b) 'The semantics of DATR' in A. G. Cohn (eds.), *Proceedings of the Seventh Conference of the Society for the Study of Artificial Intelligence and Simulation of Behavior (AISB),* Pitman/Morgan Kaufmann, London, pp. 79–87.

Flickinger, D. (1987) *Lexical Rules in the Hierarchical Lexicon,* PhD thesis, Stanford University.

Flickinger, D., C. Pollard and T. Wasow (1985) 'Structure sharing in lexical representation', *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics (ACL-85),* University of Chicago, pp. 262–268.

Flickinger, D. and J. Nerbonne (1992) 'Inheritance and complementation: A case study of *easy* adjectives and related nouns', *Computational Linguistics, vol.18.3,* 269–310.

Hayes, B. (1986) 'Inalterability in CV Phonology', *Language, vol.62 (2),* pp321–351.

Kaplan, R. (1987) 'Three seductions of computational psycholinguistics' in P. Whitelock, H. Somers, P. Bennett, R. Johnson, and M. McGee Wood (eds.), *Linguistic Theory and Computer Applications,* Academic Press, London, pp. 149–188.

Kasper, R. T. and W. C. Rounds (1986) 'A logical semantics for feature structures', *Proceedings of the 24th Annual Conference of the Association for Computational Linguistics,* Columbia University, pp. 235–242.

Kreiger, H. and J. Nerbonne (1993) 'Feature-based inheritance networks for computational lexicons' in E. J. Briscoe, A. Copestake and V. de Paiva (eds.), *Inheritance, Defaults and the Lexicon,* Cambridge University Press, Cambridge, England, pp. 90–136.

Lascarides, A. and A. Copestake (1995) 'The Pragmatics of Word Meaning', *Proceedings of the Semantics and Linguistic Theory (SALT5),* University of Texas at Austin.

Lascarides, A. and N. Asher (1993) 'Temporal Interpretation, Discourse Relations and Common Sense Entailment', *Linguistics and Philosophy, vol.16,* 437–493.

Lascarides, A., E.J. Briscoe, N. Asher and A. Copestake (in press) 'Order Independent and Persistent Typed Default Unification', *Linguistics and Philosophy,*

Reiter, R. (1980) 'A Logic for Default Reasoning', *Artificial Intelligence, vol.13,* 81–132.

Russell, G., J. Carroll and S. Warwick (1991) 'Multiple default inheritance in a unification-based lexicon', *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL-91),* Berkeley, California, pp. 215–221.

Shieber, S. M. (1986) *An introduction to unification-based approaches to grammar,* CSLI Lecture Notes 4, Stanford CA.

Vossen, P. and A. Copestake (1993) 'Untangling definition structure into knowledge representation' in E. J. Briscoe, A. Copestake and V. de Paiva (eds.), *Inheritance, Defaults and the Lexicon,* Cambridge University Press, Cambridge, England, pp. 246–274.

Young, M. and W. Rounds (1993) 'A logical semantics for nonmonotonic sorts', *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL-93),* Columbus, Ohio, pp. 209–215.