Order Independent and Persistent Typed Default Unification

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

Ted Briscoe
Rank Xerox European Research Centre, *and*
Computer Laboratory,
University of Cambridge,
New Museums Site,
Pembroke Street,
Cambridge, CB2 3QG,
England, UK
ejb@cl.cam.ac.uk

Nicholas Asher
IRIT, Université Paul Sabatier,
Toulouse, *and*
Department of Philosophy,
University of Texas at Austin,
Austin, Texas 78759,
USA
asher@bertie.la.utexas.edu

Ann Copestake
Computer Laboratory,
University of Cambridge, *and*
CSLI,
Stanford University,
Stanford, CA 94305,
USA
aac@csli.stanford.edu

August 19, 1995

Abstract

We define an order independent version of default unification on typed feature structures. The operation is one where default information in a feature structure typed with a more specific type, will override default information in a feature structure typed with a more general type, where specificity is defined by the subtyping relation in the type hierarchy. The operation is also able to handle feature structures where reentrancies are default. We provide a formal semantics, prove order independence and demonstrate the utility of this version of default unification in several linguistic applications. First, we show how it can be used to define multiple orthogonal default inheritance in the lexicon in a fully declarative fashion. Secondly, we show how default lexical specifications (introduced via default lexical inheritance) can be made to usefully 'persist beyond the lexicon' and interact with syntagmatic rules. Finally, we outline how persistent default unification might underpin default feature propagation principles and a more restrictive and constraint-based approach to lexical rules.

1 Introduction

The utility of default feature specifications has been argued for in the domain of feature propagation and specification in syntactic theory (e.g. Gazdar, 1987; Shieber, 1986b), the analysis of gapping constructions (Kaplan, 1987), and most frequently in inheritance-based accounts of lexical organisation (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, 1986a and others). In this context, a frequent motivation for allowing default inheritance in the lexicon is to capture phenomena of blocking, or the overriding of regularities by subregularities (e.g. Calder, 1989; Copestake and Briscoe, 1992; Briscoe *et al.*, 1993). Approaches to formalising and implementing inheritance-based lexicons include utilising object-oriented programming techniques (Daelemans, 1987; Flickinger *et al.*, 1985), knowledge representation languages (Pustejovsky and Boguraev, 1993; Flickinger, 1987), path-based inheritance (Evans and Gazdar, 1989a,b), overwriting templates (Karttunen, 1986; Shieber, 1986a) and various definitions of default unification (Copestake, 1993; Russell *et al.*, 1991, 1993). None of these approaches achieves the perspicuity and declarativity of subsumption-based approaches to non-default inheritance (e.g. Carpenter, 1992), since in all these accounts inheritance must be carried out in a predetermined order.

In order to provide an account of defaults which is more closely integrated with the usual monotonic operations on feature structures (FSs), it is natural to assume a notion of default unification. Most definitions have followed Kaplan (1987) in assuming that this is an asymmetric binary operation, with one default FS being regarded as adding consistent information to a non-default FS. The result is taken to be either a single FS or a disjunction of FSs. Carpenter (1993) reviews extant definitions of default unification and concludes that none provide a good underpinning for a perspicuous account of default lexical inheritance, because an inheritance hierarchy must be separately defined for which inheritance order must be stipulated, since default unification is not associative under any of these definitions. Apart from this, individual specifications of defaults which pertain to a class cannot be evaluated incrementally with an asymmetric operation because the information contributed by earlier defaults becomes non-default. Furthermore, non-associativity makes default unification unsuitable for use in situations where the application of defaults cannot be circumscribed. Order independence is not an essential property for operations on static hierarchies, although it is highly desirable, but it is necessary for applications where the order in which information can be accumulated is not predefined.

In this paper, we address these problems by defining a symmetric, order independent version of default unification on typed feature structures. Order independence is one way of allowing a series of default unifications to be evaluated incrementally, and so the order in which information can be accumulated need not be predefined. The operation will extend previous work on default unification (e.g. Bouma 1990, 1992; Copestake, 1993; Carpenter, 1993; Russell *et al.*, 1991, 1993) in that, following Young and Rounds (1993) we mark default information explicitly in feature structures rather than treating default unification as an asymmetric operation on a defeasible FS and indefeasible FS. We call these FSs, where default information is explicitly marked as such, *default* FSs (or DFSs). In DFSs, the defeasible information, if it survives unification, persists as defeasible information, via the explicit marking. So we term this approach 'persistent' default unification.

Young and Rounds (1993) define a version of persistent default unification, and provide a semantics via Default Logic (Reiter 1980). The operation is order independent, but it has two limitations which restrict its range of linguistic application. First, the operation does not permit more specific default information to override less specific default information. Secondly, the operation is not defined over FSs that incorporate default reentrancies. As

Young and Rounds choose to formalise default unification within Default Logic, the most straightforward way of extending their definition to one which prioritises defaults, would be to re-encode their definition in the extensions of default logic that are designed for prioritisation. But all these extensions deploy one of the following two strategies. Either the *order* of the application of default rules to the premises is constrained extralogically (e.g. Konolige, 1988), or prioritisation is introduced by translating the premises into the logic in a context-sensitive way. For example, if we have a class **b** which is more specific than a class **a**, and **a** defeasibly implies ϕ while **b** defeasibly implies the incompatible information ψ, and we wish to unify two FSs of classes **a** and **b**, then, and *only* then, we add the formula $b \rightarrow \neg\phi$ into the translation of the premises. In this manner the order in which defaults concerning **a** or the defaults concerning **b** are considered will not affect the result (e.g. Brewka, 1991).

Thus, extending Young and Rounds' operation to prioritise defaults via extending Default Logic, will require specifications on the order in which the default information in the two feature structures being unified is considered, or it will considerably complicate the translation of the FSs into the logic. Conditional logics for nonmonotonic reasoning have proved useful exactly because the prioritisation of defaults is defined by the axioms of the logic, and thus the priorities follow from the semantics of the defaults themselves (e.g., Boutilier 1992, Delgrande 1988, Asher and Morreau 1991, Morreau 1992). So conditional logics don't have to specify an order in which default rules are applied, nor do they have to invoke a context-sensitive translation. We will therefore use a conditional logic rather than an extension of default logic to prioritise defaults.

In addition, we also extend persistent default unification to *typed* default feature structures (TDFSs) and to TDFSs containing specifications of defeasible reentrancy. We provide a translation of TDFSs into a conditional logic, define the default unification operation via the logical consequence relation of this logic, and prove order independence via this logical consequence relation. We also discuss the computational complexity of default unification in the framework we develop. We motivate allowing default specification to 'persist beyond the lexicon' in order to elegantly encode defeasible lexical semantic information and to support seamless interaction with principles of discourse interpretation. However, we begin with constraint-based approaches to lexical organisation, motivating the utility of order-independent and persistent typed default unification as the operation underpinning (default) inheritance in the lexicon.

2 Inheritance-based Lexicons and Default Unification

Recent unification/constraint-based accounts of the lexicon have largely moved away from a conception of the lexicon as an unstructured list of lexical entries towards a hierarchical or network, inheritance-based view in which lexical entries are defined via cross-classification of their similarities (see e.g. Pollard and Sag, 1987). Under this view the lexicon is still seen by the syntagmatic component as a list of lexical entries (represented as FSs), but generalisations concerning their structure are captured by the inheritance network. For example, we may wish to define lexical signs, intransitive verbs and monadic predicates, as in Figure 1. We can specify that classes are in a partial order, illustrated as a Hasse diagram in Figure 2, so that there is a homomorphism between the partial order and the constraints on the classes. Under these conditions, the constraint specifications shown as AVMs in Figure 1 can be treated as being monotonically inherited and the constraint on a class can be calculated by unifying the constraints on its parents with its own constraint specification. The constraint on **monadic-**

$$
\begin{array}{lcl}
\textbf{sign} & \rightarrow & \left[\text{PHON} : \textbf{orth} \right] \\[6pt]
\textbf{intrans-verb} & \rightarrow & \textbf{sign} \wedge \left[\text{SYN} : \left[\begin{array}{l} \text{CAT} : \textbf{verb} \\ \text{SUBJ} : \left[\text{SYN} : \left[\text{CAT} : \textbf{np} \right] \right] \end{array} \right] \right] \\[12pt]
\textbf{monadic} & \rightarrow & \textbf{sign} \wedge \left[\begin{array}{l} \text{SYN} : \left[\text{SUBJ} : \left[\text{SEM} : \boxed{1} \right] \right] \\ \text{SEM} : \left[\begin{array}{l} \text{PRED} : \textbf{pred} \\ \text{ARG1} : \boxed{1} \end{array} \right] \end{array} \right] \\[12pt]
\textbf{monadic-intrans-verb} & \rightarrow & \textbf{monadic} \wedge \textbf{intrans-verb}
\end{array}
$$

Figure 1: FS Constraints

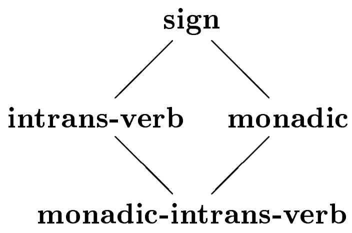


Figure 2: Schematic partial inheritance network

intrans-verb is thus the AVM shown in Figure 3.

The description of *sleep* can simply state that it is a **monadic-intrans-verb** with idiosyncratic values for phonology and semantics. Thus unifying the constraint for **monadic-intrans-verb** with the lexical description will allow us to construct the FS for *sleep*. As unification is an associative operation, partial orders / Hasse diagrams can be interpreted algebraically or logically and no 'extralogical' ordering or procedurality is being smuggled into the theory (e.g. Carpenter, 1992; de Paiva, 1993; Smolka and Aït-Kaci, 1988; Zajac, 1993). Classes can be seen as simply being part of the lexical description language, however, in what follows we assume a typed feature structure framework where the classes are types (or sorts in the terminology of Pollard and Sag (1987)) which label FS nodes. Therefore, types are an integral part of the feature structure language. Nevertheless, most of our discussion of

$$
\left[\begin{array}{l}
\textbf{monadic-intrans-verb} \\
\text{PHON} : \textbf{orth} \\
\text{SYN} : \left[\begin{array}{l} \text{CAT} : \textbf{verb} \\ \text{SUBJ} : \left[\begin{array}{l} \text{SYN} : \left[\text{CAT} : \textbf{np} \right] \\ \text{SEM} : \boxed{1} \end{array} \right] \end{array} \right] \\
\text{SEM} : \left[\begin{array}{l} \text{PRED} : \textbf{pred} \\ \text{ARG1} : \boxed{1} \end{array} \right]
\end{array} \right]
$$

Figure 3: Constraint on **monadic-intrans-verb** represented as an AVM

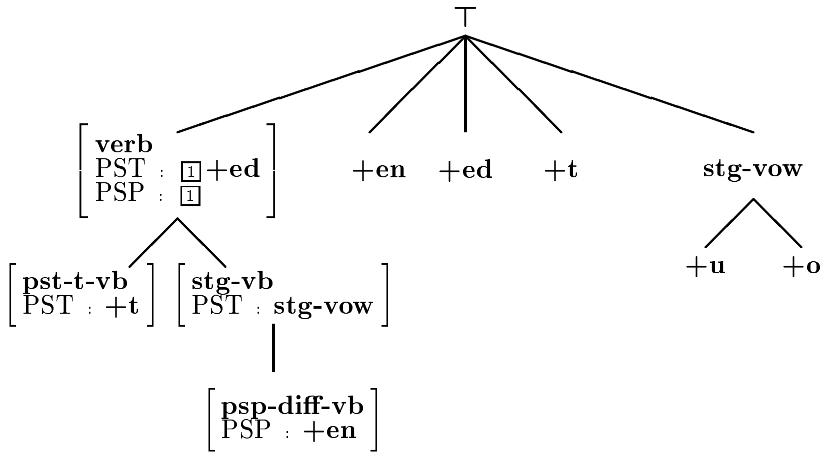


Figure 4: Partial Default Inheritance Hierarchy

inheritance is applicable to both typed and untyped frameworks.

Further abbreviation and generalisation can be obtained if the inheritance network is interpreted in a default fashion. Figure 4 gives a simple example of an inheritance hierarchy based on default inheritance. This data does not provide a convincing motivation for introducing the considerable extra machinery required to formalise default inheritance as opposed to a monotonic subsumption hierarchy — this can be found in the works cited above and in §4 and §6. However, it does suffice to illustrate the weaknesses of current definitions of default unification regarded as the operation underlying default inheritance. The intended interpretation of Figure 4 is that elements in bold represent types, with most general types at the top; thus **psp-t-vb** is a subtype of **verb**. The FSs in AVM notation associated with types represent *default constraint specifications* associated with such types. Furthermore, there is a subsumption ordering on values as indicated on the right with **+u** being an atomic leaf type which is more specific than **stg-vow**. PST encodes the past tense suffixes of English verbs. PSP encodes their past and passive participle suffixes. The full specification of **verb** will include much more information appropriate to regular verbs and each subtype will inherit this information, unless it is explicitly overridden by a specification on that subtype. Thus **psp-t-vb**s are specified to be identical to **verb**s, and thus fully regular except in their specification of the past tense suffix. Since, by default, PST is reentrant with PSP on **verb**, all the classes are meant to inherit this reentrancy constraint, except **psp-diff-vb**s where it is overridden. Thus *walk, walked* can simply be specified as **verb**, *keep, kept* as **pst-t-vb**, *strike, struck* as **stg-vb**, and *break, broke, broken* as **psp-diff-vb**.[1]

The assumption behind the use of default unification in inheritance hierarchies is that it is used instead of monotonic unification to construct the constraint on a class from its default constraint specification and the constraints on its parents. Thus the FS associated with **pst-t-verb** would be default unified with that of **verb**. However, as it stands, this example is incoherent. The FS associated with **verb** specifies a defeasible value and a defeasible reentrancy between two attributes. An operation of default unification defined to combine this information with the conflicting value specification on **pst-t-vb** is problematic because

[1]We intend that the lexical hierarchy should interact with lexical rules which relate past tense and participle forms of verbs to base forms by changing the specification of VFORM values and phonologically adjoining the value of PSP or PST to the stem. Pollard and Sag (1987) describe this approach to lexical rules and Bird and Klein (1994) describe how morphophonological operations can be characterised in a constraint-based formalism.

$$\begin{aligned}\begin{bmatrix}\text{PST} : \mathbf{+t} \\ \text{PSP} : \top\end{bmatrix} \overset{<}{\sqcap} \begin{bmatrix}\text{PST} : \boxed{1}\ \mathbf{+ed} \\ \text{PSP} : \boxed{1}\end{bmatrix} &= \left\{ \begin{bmatrix}\text{PST} : \mathbf{+t} \\ \text{PSP} : \mathbf{+ed}\end{bmatrix}, \begin{bmatrix}\text{PST} : \boxed{1}\ \mathbf{+t} \\ \text{PSP} : \boxed{1}\end{bmatrix} \right\} \quad \text{(credulous)} \\ &= \begin{bmatrix}\text{PST} : \mathbf{+t} \\ \text{PSP} : \top\end{bmatrix} \quad \text{(skeptical)}\end{aligned}$$

Figure 5: Indeterminacy of Default Unification

it is not clear which aspects of the defeasible information on the supertype constraint should survive – the reentrancy or the value or neither. The situation is represented in Figure 5 where $\overset{<}{\sqcap}$ denotes default unification of an indefeasible FS with a defeasible FS, *indefeasible* FS $\overset{<}{\sqcap}$ *defeasible* FS (or equivalently here of a subtype FS with a parent type FS). (We use $\top$ as equivalent to unspecified.) The results shown are those using the definitions proposed by Carpenter (1993): the first (credulous) result is the set of maximally informative default extensions of the indefeasible FS, the second (skeptical) result is the generalisation of this set.[2] In Figure 4 it is now possible to see that different choices are required at different points: in all cases except **psp-diff-vb** we want the original defeasible reentrancy between PST and PSP to survive; with **psp-diff-vb** we want the reentrancy to be rejected in favour of the distinct values. Clearly, if we are to adopt a single consistent definition of default unification we must alter the hierarchy of Figure 4 by explicitly specifying that the reentrancy is retained in **pst-t-vb** and **stg-vb**. One important design consideration in the definition of a useful default unification operation is the computational complexity of the resultant system. For this reason, in the case of such indeterminacies, we would prefer to adopt Carpenter's (1993) skeptical approach and define the result as the skeptical extension of the indefeasible or more specific information with (more general) default information; thus, avoiding indeterminacy in the outcome.

But there is a general problem with the asymmetric definitions of default unification considered by Bouma (1990, 1992), Carpenter (1993), Copestake (1993) and Russell *et al.* (1991, 1993), because defeasible and indefeasible information in FSs is not kept distinct. Instead, in each application of the operation one argument FS is treated as entirely defeasible and the other as entirely indefeasible, and the result is a 'normal' FS or set of FSs as in Figure 5. It is straightforward to see that none of these definitions is commutative — if we switch the order of the arguments in Figure 5 the result will be different because we also switch the defeasibility of the information in each FS. What is less obvious is that the results are also not associative: Figure 6 shows one example which illustrates this for Carpenter's credulous and skeptical operations. Thus, if we use a definition of this type to implement default inheritance in a lexical hierarchy, we either have to define the result as the values obtained by considering all possible orders (or their generalisation), which is clearly unattractive both for computational tractability and perspicuity, or we have to make an additional procedural stipulation of inheritance order. For example, we would, in general, obtain different results if we proceeded 'top-down', composing successively more specific default constraints, than if we proceed 'bottom-up' adding successively more general default specifications (see Carpenter, 1993 for further discussion).

Young and Rounds (1993) provide a definition of default unification of FSs in which defeasible information is explicitly marked. This enables them to provide a definition which is

[2] The results given by some alternative definitions are discussed in Copestake (1993). Carpenter's definitions treat path equality (reentrancy) information as having equal status with path value information, but there are alternative options where one or other is treated as having priority.

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

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

Figure 6: Non-associativity of Asymmetric Default Unification

both commutative and associative. However as we've mentioned, their approach is limited by two factors; firstly, their definition doesn't deal with reentrancy, so their operation would not support the overriding of reentrancy in Figure 4 on **psp-diff-vb**; secondly, they do not prioritise defaults. Thus, their definition is limited to cases of indefeasible values overriding conflicting defeasible ones. This would be adequate to capture the overriding of PSP on **stg-vb** in Figure 4, but not to allow this value to be overridden by that on **psp-diff-vb**, for example. We have argued in favour of extending their work by formalising default unification in a conditional logic, rather than an extension of Default Logic. This will enable us to introduce prioritisation of defaults into the operation, without constraining the order in which default information is accumulated into the result of unifying two FSs.

3 Persistent Default Unification

In this section we present a definition of persistent default unification of typed feature structures (TFSs) which is order independent, allows for default reentrancy and has a built-in notion of specificity. We begin by defining typed default feature structures (TDFSs), which are more complex than normal FSs in order to incorporate both default and non-default information, and then informally describe persistent default unification ($\overset{<>}{\sqcap}$) and illustrate its behaviour with examples. To formalise $\overset{<>}{\sqcap}$ we provide a translation function between TDFSs and formulae of a conditional logic in §5, and axiomatise the conditional logic so that the operation $\overset{<>}{\sqcap}$ has the desired properties.

3.1 Typed default feature structures

We assume a theory of TFSs which is similar to that of Carpenter (1992), Copestake *et al.* (1993) and Zajac (1993) in which types label nodes in FSs (shown in bold in AVM diagrams but omitted where irrelevant). The definition of the type hierarchy $\langle \mathsf{Type}, \sqsubseteq \rangle$ is equivalent to that in Carpenter (1992). However, we use a convention where more specific types are described as being lower in the hierarchy, thus $\top$ is the most general type, and $\bot$ indicates inconsistency. The hierarchy is a lattice, so any set of types will have a unique greatest lower bound or meet. Monotonic unification ($\sqcap$) of a set of TFSs is defined to result in an FS which has a type equal to the meet of the types of the unified structures and is usually said to fail if this is $\bot$. However, in what follows, we will allow FSs to contain nodes which are typed $\bot$. We assume all TFSs are acyclic and we treat all values as intensional; that is, there can be more than one instance of any type in a TFS (see Carpenter 1992 for details). Treating values this way is more general than treating them as extensional: persistent default unification can handle both kinds of structures. On the other hand, we leave open how the operation would have to be extended to handle cyclic structures.

To define order independent persistent default unification of TFSs we introduce a notation for TDFSs which, informally, contains a 'double' TFS in which the left-hand FS specifies what is indefeasible and the right-hand FS what is defeasible. For instance, the constraint associated with **verb** could be represented as (1), which makes explicit the fact that it is the reentrancy specification and value on PST which are default, rather than the presence of the attributes PST and PSP.

$$(1) \qquad \begin{bmatrix} \mathbf{verb}^1 \\ \text{PST} : \top \\ \text{PSP} : \top \end{bmatrix} \Big/ \begin{bmatrix} \mathbf{verb}^1 \\ \text{PST} : \boxed{1}\,\mathbf{+ed} \\ \text{PSP} : \boxed{1} \end{bmatrix} \quad /\{\langle \text{PST}, \{\langle \text{+ed}, \text{verb}\rangle\}\rangle, \langle \text{PSP}, \{\langle \text{+ed}, \text{verb}\rangle\}\rangle\}$$

The third element in the structure is a *tail*. Tails record the information we need from the history of default unifications a TDFS has undergone, in order to guarantee order independence of the operation. For each path, the tail contains *value-specificity* pairs, which record a default *value* that appeared on that path in a TDFS that was used to build the TDFS being considered, together with the type of the root node of that TDFS. The position of this root type in the type hierarchy (which we assume to be a finite bounded complete partial order, as defined in Carpenter (1992; 11–13)) records the *specificity* of the default information. In general, during a series of default unifications, tails will record the values on paths of original TDFSs, together with the specificity of those TDFSs. For example, if a TDFS has a tail of the form:

$$\{\langle \pi, \{\langle a, t_1\rangle, \langle b, t_2\rangle\}\rangle, \langle \pi', \{\langle c, t_3\rangle, \langle d, t_4\rangle\}\rangle\}$$

then this means that to produce this TDFS we unified: a TDFS that had a root type t_1 and defeasible path $\pi : a$; a TDFS that had a root type t_2 and defeasible path $\pi : b$; a TDFS that had a root type t_3 and defeasible path $\pi' : c$; and a TDFS that had a root type t_4 and defeasible path $\pi' : d$. The relative order of t_1, t_2, t_3 and t_4 in the type hierarchy determines the relative specificity of the paths $\pi : a$, $\pi : b$, $\pi' : c$ and $\pi' : d$ respectively. As will become clear below, keeping track of this information is the cost we pay for being able to define default unification as an order independent operation. We will refer to the initial specifications of defaults as the initial knowledge base (KB): for any FS in the initial KB the tail will just contain the path value information present which is defeasible; i.e., it appears in the defeasible FS, but is strictly more specific than the indefeasible FS. We may refer to such information as *strictly default*. Note that we recorded the values on both the equivalent paths PST and PSP in the tail of (1). Path equivalences themselves are not recorded in tails, however. A fourth element in a TDFS is a tag shown as a superscript on the root types of the FSs. This is used to identify TDFSs, to aid the translation to conditional logic. We will normally omit indices in this section, since they do not affect the results of the default unification operation.

We can make some convenient abbreviations, so that the 'double' AVM notation can be factored across individual paths into a single AVM representation where we have *Indefeasible Value*/*Defeasible Value* which in turn can be abbreviated to *Indefeasible Value* where *Indefeasible Value* is the same as *Defeasible Value*, and /*Defeasible Value* where $\top$/*Defeasible Value*, without ambiguity. We can also omit the tail, when it can be determined from the default FS. Thus (1) can be represented as (2).

$$(2) \qquad \begin{bmatrix} \mathbf{verb} \\ \text{PST} : /\boxed{1}\,\mathbf{+ed} \\ \text{PSP} : /\boxed{1} \end{bmatrix}$$

Using this notation, our new version of the default inheritance hierarchy introduced in the previous section is shown in Figure 7.

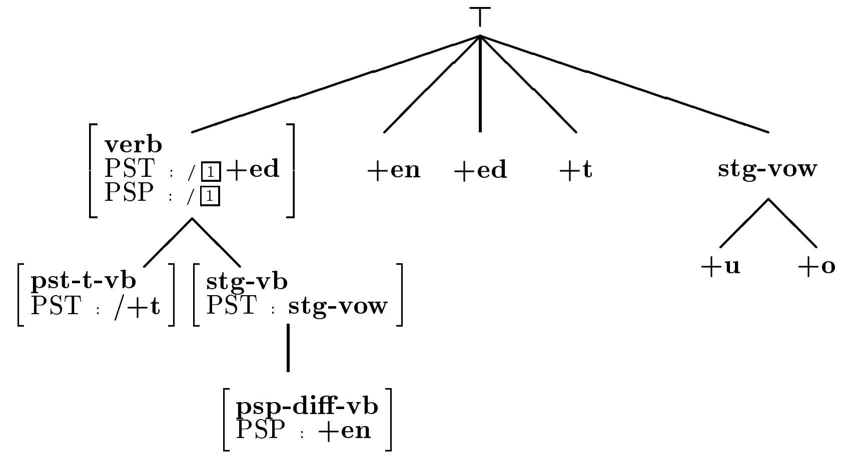


Figure 7: Partial Default Inheritance Hierarchy (new version)

In the formal definition of TDFS below, we ensure that in a well-formed TDFS the indefeasible specification subsumes the defeasible specification; that is, defaults *add* information to what is known indefeasibly. However, it does not specify any relationship between the FSs and the tail. This relationship in the TDFSs for the initial KB is conventionally defined as above, and the way tails are merged during unification will be defined by the unification operation itself, rather than via the definition of TDFSs. This means there are some TDFSs that are well-formed, but which would never be in an initial KB, nor constructed via unification from those TDFSs in the initial KB.

Firstly, we reiterate Carpenter's (1992) definition of a type hierarchy, on which the definition of TDFSs rests.

Definition 1 Type Hierarchy
A type hierarchy is a finite bounded complete partial order $\langle \mathsf{Type}, \sqsubseteq \rangle$.

This guarantees that any set of types will have a unique most general unifier, which will be $\perp$ in the case of inconsistent types. We now define tails, and then TDFSs.

Definition 2 Tails
A tail is a set of pairs, where

- *The first member of the pair is a sequence of features (i.e. a path);*
- *the second member of the pair is a set of pairs of types.*

Definition 3 Typed Default Feature Structure
A typed default feature structure defined on a set of features Feat*, a set of types* Type *in a type hierarchy* $\langle \mathsf{Type}, \sqsubseteq \rangle$ *and a set of indices* N *is a tuple* $\langle \mathcal{T}, i, Q, \mathcal{R}, \delta, \theta \rangle$*, where:*

- $\mathcal{T}$ *is a tail.*
- $i \in N$ *is the tag of the* TDFS.
- Q *is a finite set of nodes,*
- $\mathcal{R} \in Q \times Q$
  ($\mathcal{R}$ are the root nodes of the indefeasible and defeasible TFSs*).*

- $\theta : Q \longrightarrow \mathsf{Type}$ *is a partial typing function.*
- $\delta : Q \times \mathsf{Feat} \longrightarrow Q$ *is a partial feature value function,*

such that 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 *such that* $\delta \ldots (\delta(n, F_1), F_2) \ldots F_k) = n$
4. $\theta(n_2) = \theta(n_1)$
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* $\delta(\ldots(\delta(n_1, F_1), F_2) \ldots F_k) = n$*, then* $\delta(\ldots(\delta(n_2, F_1), F_2) \ldots F_k) = m$*; and*
   (b) $\theta(m) \sqsubseteq \theta(n)$

   (1, 2 and 3 ensure that we have two DAG*s rooted at* n_1 *and* n_2*, and 4 and 5 ensure that defeasible information is at least as specific as indefeasible information).*

In the above definition, there is an FS containing the indefeasible information rooted at n_1, and another containing the defeasible information rooted at n_2. Constraints 1, 2 and 3 on δ ensure that we have two rooted DAGS. Constraints 4 and 5 ensure that the defeasible information entails the indefeasible information (as long as the root of the defeasible part isn't $\bot$, in which case nothing consistent can be said about what normally holds of the TDFS). Note in particular that constraint 5a ensures any indefeasible reentrancy is also present in the defeasible DAG. There are no constraints on the relation between a tail and the DAGs in the *definition* of TDFSs, but we will specify below how tails are constructed, as part of the logic of default unification.

3.1.1 A detailed example

(3) is a TDFS, which is indexed 1 (as shown by the superscript on the root type).

(3)
$$\begin{bmatrix} \mathbf{t}^1 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{a} \end{bmatrix} \\ \mathrm{H} : \mathbf{b} \end{bmatrix} \Bigg/ \begin{bmatrix} \mathbf{t}^1 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : \boxed{2}\,\mathbf{a} \\ \mathrm{H} : \boxed{2} \end{bmatrix} \\ \mathrm{H} : \mathbf{c} \end{bmatrix} \Bigg/ \{\langle F \cdot H, \{\langle a, t\rangle\}\rangle, \langle H, \{\langle c, t\rangle\}\rangle\}$$

Or using the abbreviated AVM notation, we obtain (4):

(4)
$$\begin{bmatrix} \mathbf{t}^1 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{a}/\boxed{2} \\ /\mathrm{H} : \boxed{2} \end{bmatrix} \\ \mathrm{H} : \mathbf{b}/\mathbf{c} \end{bmatrix}$$

A tuple that describes this TDFS is $\langle \mathcal{T}, 1, Q, (n_1, n_5), \delta, \theta \rangle$, where:

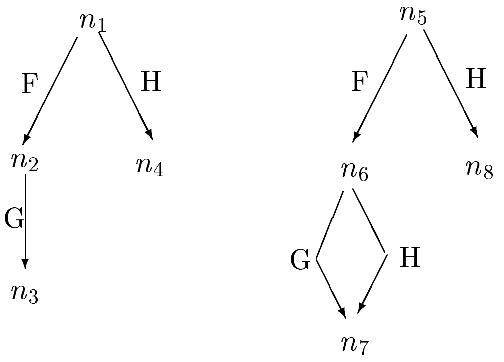
1. $\mathcal{T} = \{\langle F \cdot H, \{\langle a, t\rangle\}\rangle, \langle H, \{\langle c, t\rangle\}\rangle\}$

2. $Q = \{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8\}$

3. $\delta(n_1, F) = n_2$, $\delta(n_2, G) = n_3$, $\delta(n_1, H) = n_4$, $\delta(n_5, F) = n_6$, $\delta(n_6, G) = n_7$, $\delta(n_6, H) = n_7$, $\delta(n_5, H) = n_8$

4. $\theta(n_1) = t$, $\theta(n_3) = a$, $\theta(n_4) = b$, $\theta(n_5) = t$, $\theta(n_7) = a$, $\theta(n_8) = c$

The two rooted DAGS can be drawn pictorially as follows:



Note that for the TDFS to be well defined, c must be a subtype of b.

This TDFS could have been part of the initial KB, since the value-type pairs in the tails match exactly the strictly defeasible information in the defeasible FS. If the tail were totally unrelated to the information in the FSs—for example, it specified values on paths that don't appear in the FSs—then it is a well-defined TDFS, but it would never be generated from the initial KB via the unification operation $\overset{<>}{\sqcap}$ defined below. We abbreviate the part of the tail for a path π, for a TDFS tagged 1, to π^1. For example $F \cdot H^1 = \{\langle a, t\rangle\}$ in this example.

3.2 An informal definition of $\overset{<>}{\sqcap}$

We have assumed requirements for default unification which correspond to those in Copestake (1993), adapted because we are assuming persistent defaults with the addition of order independence and specificity. We want our definition of $\overset{<>}{\sqcap}$ to meet the following criteria:

1. Default information must always be consistent with non-default information.

2. Default unification behaves like monotonic unification in the cases where there are no conflicts in the default information.[3]

3. Default unification never fails when the non-default information is consistent.

4. Default unification returns a single result, deterministically.

5. Default values can be given a specificity ordering, such that defaults associated with a more specific class take priority over those associated with a more general class.

6. Default unification can be described using a binary, order independent operation.

[3] As we will see in §5.5, the definition of conflict is more complex because of tails.

The first five criteria can be understood to apply globally to a set of TDFSs which are to be unified. In contrast, the last criterion is necessary because we assume that we cannot, in general, expect to have all this information available at once. In principle, this would be possible for a static inheritance hierarchy, although in practice it would seriously constrain possible implementations and would make the operation much less perspicuous than monotonic unification. Furthermore, defaults have utility in applications where there is no definite point at which it is possible to decide that no further relevant information will apply.

But 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 $\overset{<>}{\sqcap}$ so that it has all the desired properties.

We now informally describe the operation which we define logically in section 5. We also prove that this operation meets the above criteria in that section.

Let $F_1 = I_1/D_1/T^1$ be a TDFS, and so I_1 is the indefeasible TFS, D_1 the defeasible TFS and T^1 the tail. Similarly for F_2. Then $F_{12} =_{def} I_{12}/D_{12}/T^{12} =_{def} F_1 \overset{<>}{\sqcap} F_2$ is calculated as follows:[4]

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. Default Reentrancies and Paths:
   They all survive in D_{12}.
   We use $\approx$ to indicate reentrancy, and $\pi \approx_{D_1} \pi'$ means $\pi \approx \pi'$ is true in D_1. So if $\pi \approx_{D_1} \pi'$ or $\pi \approx_{D_2} \pi'$, then $\pi \approx_{D_{12}} \pi'$. If adding all the reentrancies in D_1 and D_2 gives a cycle, then as in normal unification, D_{12} is $\perp$ and we're done. If not, then note that paths also survive, because for any path π, $\pi \approx \pi$; the defeasible values are then calculated according to step 4 below.

3. Tails:
   We calculate T^{12} as follows:
   For each path π which has a value in T^1 or T^2, π^{12} is the union of π'^1 and π'^2, for all π' that are reentrant with π in D_{12}.

   $$\pi^{12} = \{\pi'^1 : \langle\pi\rangle \approx_{D_{12}} \langle\pi'\rangle\} \cup \{\pi'^2 : \langle\pi\rangle \approx_{D_{12}} \langle\pi'\rangle\}$$

   Note we already know which paths are reentrant because we've done step 2. Also note that $\pi^{12} \subseteq \pi^1 \cup \pi^2$, because $\pi \approx \pi$.

4. Default Values:
   These are calculated via the tail T^{12}, and each path π is considered separately.
   For any path π which has a value in D_1 or D_2, the default value on π in D_{12} is the unification of the following:

[4] Note that we sometimes abbreviate the index $1 \wedge 2$ to 12, when there's no confusion.

Indef $\mathit{Indef}^{12}_{\pi}$ is the unification of the indefeasible values in I_{12} on all the paths that are reentrant with π in D_{12}
$\mathit{Indef}^{12}_{\pi} = \sqcap\{\phi_{\pi'} : \langle\pi'\rangle =_{I_{12}} \phi_{\pi'} \wedge \langle\pi\rangle \approx_{D_{12}} \langle\pi'\rangle\}$
Where $\langle\pi'\rangle =_{I_{12}} \phi_{\pi'}$ means that I_{12} has a π' path, and the value on this path is $\phi_{\pi'}$.

Spec Spec^{12}_{π} is the set of values in π^{12} that are associated with the maximally specific types, and are also compatible with $\mathit{Indef}^{12}_{\pi}$
$\mathit{Spec}^{12}_{\pi} = \{\phi : \langle\phi, t_{\phi}\rangle \in \pi^{12} \text{ and } \forall\langle\phi', t_{\phi'}\rangle \in \pi^{12}, t_{\phi'} \not\sqsubset t_{\phi} \text{ and } \phi \sqcap \mathit{Indef}^{12}_{\pi} \neq \bot\}$

Comp Comp^{12}_{π} is the set of values in π^{12} that are individually compatible with $\mathit{Indef}^{12}_{\pi}$ and each value of Spec^{12}_{π}
$\mathit{Comp}^{12}_{\pi} = \{\phi : \langle\phi, t_{\phi}\rangle \in \pi^{12} \text{ and } \forall v \in \mathit{Spec}^{12}_{\pi}, v \sqcap \phi \neq \bot \text{ and } \phi \sqcap \mathit{Indef}^{12}_{\pi} \neq \bot\}$

Some explanation of the steps 2, 3 and 4 are in order. First, step 2 ensures all reentrancies and paths survive, regardless of the indefeasible and defeasible values involved.

It is possible to extend the logic so that defeasible reentrancies can be overridden, but this comes at a cost in computational complexity, as discussed in §4.1. The operation that extends $\overset{<>}{\sqcap}$ in this way is defined in Lascarides and Copestake (1995). Second, step 3 merges tails so that the π and π' parts of the new tail are equal if π' is reentrant with π. We must construct new tails this way, so that the defeasible values as defined in step 4 produce a well defined TDFS. If we did not add π' parts where π' is reentrant with π to the π-part of the new tail, then step 4 would not guarantee that the defeasible values on defeasibly reentrant paths were equal.

Finally, we explicate *Indef*, *Spec* and *Comp* in step 4. First, *Indef* ensures that the defeasible value on a path is at least as specific as the indefeasible value on that path. It's also at least as specific as the values on the paths with which it's defeasibly reentrant. This is essential for the results of $\overset{<>}{\sqcap}$ to be a well-defined TDFS. Without this, the defeasible values are not guaranteed to be both at least as specific as the indefeasible ones, and equal on defeasibly reentrant paths. Second, *Spec* and *Comp* use only the values from the tails, and not the values in D_1 and D_2. This feature yields order independence; this is proved in §5. *Spec* ensures values associated with most specific types survive, but only if they were compatible with the indefeasible information. If we did not ensure compatibility, then indefeasible information would not override defeasible information; rather we would get the value $\bot$ when they conflict. The set *Comp* ensures that compatible information on more general types also survives. This is necessary for both order independence, and ensuring that $\overset{<>}{\sqcap}$ reduces to $\sqcap$ when no conflicts arise. Again, this is proved in §5. It also means that values that originated from more general TDFSs than those used to construct *Spec*, which are incompatible with the values in *Spec*, do not survive. This ensures that $\overset{<>}{\sqcap}$ incorporates specificity. The more specific defaults will survive in *Spec*, and block the more general conflicting defaults from being in *Comp*.

These steps can be carried out in the order specified, and each step produces a deterministic result. Nothing in step 2, for example, is dependent on the results of step 4. Furthermore, note that one can do step 4 path by path in any order: the value calculated at one node does not affect the value of another node. The one exception is when the reentrancies in D_1 and D_2 produce a cylce. In this case, the logic in section 5 yields the result that $D_{12} = \bot$.

It is possible to illustrate the persistent default unification of any two TDFSs using this informal definition.

Defeat of Defeasible Modus Ponens
Where $t_2 \sqsubset t_1$ and $a \sqcap b = \bot$

$$\begin{bmatrix} \mathbf{t}_1 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{a} \end{bmatrix} \end{bmatrix} /\{\} \overset{<>}{\sqcap} \begin{bmatrix} \mathbf{t}_2 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : /\mathbf{b} \end{bmatrix} \end{bmatrix} \Big/ \{\langle F \cdot G, \{\langle b, t_2 \rangle\}\rangle\} \ = \begin{bmatrix} \mathbf{t}_2 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{a} \end{bmatrix} \end{bmatrix} \Big/ \{\langle F \cdot G, \{\langle b, t_2 \rangle\}\rangle\}$$

To show that the above holds, we work through each step of the definition of $\overset{<>}{\sqcap}$ in turn. We concentrate on the value on the $F \cdot G$ path; the root type of the two TDFSs in the result is calculated in a similar fashion.

1. The indefeasible $F \cdot G$ value is $a \sqcap \top = a$.

2. All defeasible paths and reentrancies survive (there are no defeasible reentrancies).

3. $F \cdot G^{12} = F \cdot G^1 \cup F \cdot G^2 = \{\langle b, t_2 \rangle\}$

4. The defeasible $F \cdot G$ value is the unification of:

   Indef a

   Spec $\{\}$ (because b is incompatible with a).

   Comp $\{\}$.

   which is a.

Specificity
The following example demonstrates the prioritisation of defaults.
Where $t_2 \sqsubseteq t_1$ and $a \sqcap b = \bot$

$$\begin{bmatrix} \mathbf{t}_1 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : /\mathbf{a} \end{bmatrix} \end{bmatrix} \Big/ \{\langle F \cdot G, \{\langle a, t_1 \rangle\}\rangle\} \ \overset{<>}{\sqcap} \begin{bmatrix} \mathbf{t}_2 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : /\mathbf{b} \end{bmatrix} \end{bmatrix} \Big/ \{\langle F \cdot G, \{\langle b, t_2 \rangle\}\rangle\}$$

$$= \begin{bmatrix} \mathbf{t}_2 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : /\mathbf{b} \end{bmatrix} \end{bmatrix} \Big/ \{\langle F \cdot G, \{\langle a, t_1 \rangle, \langle b, t_2 \rangle\}\rangle\}$$

Again, we concentrate on just the $F \cdot G$ path.

1. The indefeasible value on $F \cdot G$ is $\top \sqcap \top = \top$

2. The defeasible reentrancies and paths survive.

3. The $F \cdot G^{12}$ tail is $F \cdot G^1 \cup F \cdot G^2 = \{\langle a, t_1 \rangle, \langle b, t_2 \rangle\}$

4. The defeasible value on the $F \cdot G$ tail is
   the unification of:

   Indef $\top$

   Spec $\{b\}$ (because b is associated with a more specific type (namely t_2) than a is (namely t_1), and b is compatible with $\top$).

Comp $\{b\}$ (a is not compatible with b and so a is not in this set).

which is b.

The Nixon Diamond
We now show what happens where there is no prioritisation between conflicting defaults. Where $a \sqcap b = \bot$

$$\begin{bmatrix} \mathbf{t}_1 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : /\mathbf{a} \end{bmatrix} \end{bmatrix} \Big/ \{\langle F \cdot G, \{\langle a, t_1 \rangle\}\rangle\} \overset{<>}{\sqcap} \begin{bmatrix} \mathbf{t}_2 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : /\mathbf{b} \end{bmatrix} \end{bmatrix} \Big/ \{\langle F \cdot G, \{\langle b, t_2 \rangle\}\rangle\}$$

$$= \begin{bmatrix} \mathbf{t}_2 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : /\bot \end{bmatrix} \end{bmatrix} \Big/ \{\langle F \cdot G, \{\langle a, t_1 \rangle, \langle b, t_2 \rangle\}\rangle\}$$

Again, we concentrate just on the results on the $F \cdot G$ path.

1. The indefeasible value on $F \cdot G$ is $\top \sqcap \top = \top$
2. The defeasible paths and reentrancies all survive.
3. The $F \cdot G^{12}$ tail is $F \cdot G^1 \cup F \cdot G^2 = \{\langle a, t_1 \rangle, \langle b, t_2 \rangle\}$
4. The defeasible value on the $F \cdot G$ tail is
   the unification of:

   Indef $\top$

   Spec $\{a, b\}$
   (since both are as specific as possible)

   Comp $\{\}$
   (a is incompatible with b and b with a)

   which is $\bot$.

The value on the default node is $\bot$, indicating that nothing coherent can be said about the default information. This value could be overridden by a subsequent default unification, however, by a default $F \cdot G$ value that was associated with a more specific type than both t_1 and t_2.

Reentrancy
Since reentrancy always survives, we also get $\bot$ to be the default value when the indefeasible values on defeasibly reentrant paths are incompatible.
Where $a \sqcap b = \bot$

$$\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \mathbf{a} \\ \mathrm{G} : \mathbf{b} \end{bmatrix} \overset{<>}{\sqcap} \begin{bmatrix} \mathbf{t} \\ \mathrm{F} : /\boxed{1} \\ \mathrm{G} : /\boxed{1} \end{bmatrix} = \begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \mathbf{a}/\boxed{1}\bot \\ \mathrm{G} : \mathbf{b}/\boxed{1} \end{bmatrix}$$

We calculate the values on just the F path here; the one for G is similar.

1. The indefeasible value on F is a.
2. All default reentrancies survive; so $F \approx G$ survives.

3. The F^{12} tail is $F^1 \cup F^2 \cup G^1 \cup G^2 = \emptyset$

4. The default value on the F path is the unification of:

   Indef The indefeasible value on F *and* G
   $a \sqcap b = \perp$
   (must do this for TFSs to be well defined).

   Spec $\{\}$

   Comp $\{\}$

   which is $\perp$.

This examples clarifies the motivation for defining *Indef* in the way we do. If we did not unify the indefeasible values on the paths that are defeasibly reentrant with F, then we would not obtain a well-defined TDFS via the operation: either the defeasible values on the F and G paths would be different in spite of the defeasible reentrancy, or the defeasible value on F would not be at least as specific as the indefeasible value. Since defeasible reentrancies always survive, and indefeasible values can only get more specific with subsequent unifications, the value $\perp$ in this case could not be overridden in subsequent unifications, in contrast to the Nixon Diamond case above.

As a further example, we consider the inheritance hierarchy shown in Figure 7. We do not need to make any stipulation of top-down versus bottom-up inheritance order to be able to calculate the inherited constraint specification using this definition of default unification. The constraint on **pst-t-vb**, for example, is determined by default unification with the constraints on its supertypes.

$$\textbf{pst-t-vb} \sqsubset \textbf{verb}$$

$$1)\ \begin{bmatrix} \textbf{verb} \\ \text{PST} : /\boxed{1}\textbf{+ed} \\ \text{PSP} : /\boxed{1} \end{bmatrix} \qquad 2)\ \begin{bmatrix} \textbf{pst-t-vb} \\ \text{PST} : /\textbf{+t} \\ \text{PSP} : \top \end{bmatrix}$$

$$1 \overset{<>}{\sqcap} 2 = 2 \overset{<>}{\sqcap} 1 = \begin{bmatrix} \textbf{pst-t-vb} \\ \text{PST} : /\boxed{1}\textbf{+t} \\ \text{PSP} : /\boxed{1} \end{bmatrix} \Big/ \begin{matrix} \{\langle \text{PST}, \{\langle \textbf{+t}, \textbf{pst-t-verb}\rangle, \langle \textbf{+ed}, \textbf{verb}\rangle\}\rangle, \\ \langle \text{PSP}, \{\langle \textbf{+t}, \textbf{pst-t-vb}\rangle, \langle \textbf{+ed}, \textbf{verb}\rangle\}\rangle\} \end{matrix}$$

Inheritance hierarchies are discussed in more detail in §4.2.

So far we've shown how the definition works. We now informally demonstrate why tails are required for order independence. We do this via two more examples.

3.2.1 Keeping track of specificity

Consider another version of the hierarchy given in Figure 7, such that we have the constraint specifications shown in (5).

(5) $\textbf{psp-diff-vb} \sqsubset \textbf{stg-vb} \sqsubset \textbf{verb}$

$$1)\ \begin{bmatrix} \textbf{verb} \\ \text{PST} : /\textbf{+ed} \\ \text{PSP} : /\textbf{+ed} \end{bmatrix}\ 2)\ \begin{bmatrix} \textbf{stg-vb} \\ \text{PST} : /\textbf{+o} \\ \text{PSP} : /\textbf{+o} \end{bmatrix}\ 3)\ \begin{bmatrix} \textbf{psp-diff-vb} \\ \text{PST} : \top \\ \text{PSP} : \textbf{+en} \end{bmatrix}$$

$$1 \overset{<>}{\sqcap} 2 = \begin{bmatrix}\textbf{stg-vb}\\ \text{PST} : /\mathbf{+o}\\ \text{PSP} : /\mathbf{+o}\end{bmatrix}$$

$$(1 \overset{<>}{\sqcap} 2) \overset{<>}{\sqcap} 3 = \begin{bmatrix}\textbf{psp-diff-vb}\\ \text{PST} : /\mathbf{+o}\\ \text{PSP} : \mathbf{+en}\end{bmatrix}$$

$$1 \overset{<>}{\sqcap} 3 = \begin{bmatrix}\textbf{psp-diff-vb}\\ \text{PST} : /\mathbf{+ed}\\ \text{PSP} : \mathbf{+en}\end{bmatrix}$$

$$(1 \overset{<>}{\sqcap} 3) \overset{<>}{\sqcap} 2 = \begin{bmatrix}\textbf{psp-diff-vb}\\ \text{PST} : /\mathbf{+ed}\\ \text{PSP} : \mathbf{+en}\end{bmatrix}$$

Figure 8: Order Dependence if Specificity is Driven by the Root Type

Here, **verb** is specified to necessarily have two attributes PSP and PST both with default values **+ed**, **stg-vb** has **+o** for both attributes, and **psp-diff-vb** has **+en** for PSP. We will assume that the latter specification is indefeasible, although making it defeasible would not affect the order dependence of the result discussed below. We can also assume that the user has made no stipulation about PST for **psp-diff-vb** but that the constraint specification is well-typed, so that its value for PST is $\top$, although the results would be the same if we simply assumed PST was absent.

Suppose that, instead of having tails, we defined specificity to be driven by the root type of the current feature structure. This is a reasonable idea, since types persist in the FS (in contrast with template names in PATR-II and similar systems). Unfortunately it would result in order dependence, as shown in Figure 8. This occurs because a constraint which invokes a general value on a path is promoted in specificity: namely, the information that PST = **+ed** is promoted in specificity when unifying 1 and 3. It starts as a default on a FS of type **verb**, and becomes a default on a FS of type **psp-diff-verb**, with nothing to distinguish it from the value of PSP which was actually stipulated. Intuitively, one needs to record for each path in each TDFS, the specificity of the TDFS from which that path's value *originated.* And it must be the *original* specificity that determines which defaults survive during unification. Our use of tails to record the original specificity achieves this, as shown in Figure 9.

3.2.2 Keeping track of specificity and values

For many examples, including that just shown, we do not need such complex tails. In §5.6, we describe a sublanguage for which the information that must be retained to give order independence is just the original specificity for a given path. We could do this by associating an index with each path, represented below as an integer superscript, and having a specificity order on indices, which mirrored the specificity order on the original root types. But tails are doing more than this, since they are also recording the original values, and the next example illustrates why this is necessary for the full language.

$t_3 \sqsubset t_1 \sqsubset t_2$ $\qquad 3 \sqsubset 1 \sqsubset 2 \qquad a \sqsubset b \qquad d \sqcap b = c \qquad d \sqcap a = \bot$

(1) $\begin{bmatrix}\mathbf{t}_1^1\\ \text{F} : /\mathbf{a}^1\end{bmatrix}$ (2) $\begin{bmatrix}\mathbf{t}_2^2\\ \text{F} : /\mathbf{b}^2\end{bmatrix}$ (3) $\begin{bmatrix}\mathbf{t}_3^3\\ \text{F} : /\mathbf{d}^3\end{bmatrix}$

$$1 \stackrel{<>}{\sqcap} 2 = \begin{bmatrix} \textbf{stg-vb} \\ \text{PST} : /\textbf{+o} \\ \text{PSP} : /\textbf{+o} \end{bmatrix} \Big/ \begin{array}{l} \{\langle \text{PST}, \{\langle \textbf{+o}, \textbf{stg-vb}\rangle, \langle \textbf{+ed}, \textbf{verb}\rangle\}\rangle, \\ \langle \text{PSP}, \{\langle \textbf{+o}, \textbf{stg-vb}\rangle, \langle \textbf{+ed}, \textbf{verb}\rangle\}\rangle\} \end{array}$$

$$(1 \stackrel{<>}{\sqcap} 2) \stackrel{<>}{\sqcap} 3 = \begin{bmatrix} \textbf{psp-diff-vb} \\ \text{PST} : /\textbf{+o} \\ \text{PSP} : \textbf{+en} \end{bmatrix} \Big/ \begin{array}{l} \{\langle \text{PST}, \{\langle \textbf{+o}, \textbf{stg-vb}\rangle, \langle \textbf{+ed}, \textbf{verb}\rangle\}\rangle, \\ \langle \text{PSP}, \{\langle \textbf{+o}, \textbf{stg-vb}\rangle, \langle \textbf{+ed}, \textbf{verb}\rangle\}\rangle\} \end{array}$$

$$1 \stackrel{<>}{\sqcap} 3 = \begin{bmatrix} \textbf{psp-diff-vb} \\ \text{PST} : /\textbf{+ed} \\ \text{PSP} : \textbf{+en} \end{bmatrix} \Big/ \{\langle \text{PST}, \{\langle \textbf{+ed}, \textbf{verb}\rangle\}\rangle, \langle \text{PSP}, \{\langle \textbf{+ed}, \textbf{verb}\rangle\}\rangle\}$$

$$(1 \stackrel{<>}{\sqcap} 3) \stackrel{<>}{\sqcap} 2 = \begin{bmatrix} \textbf{psp-diff-vb} \\ \text{PST} : /\textbf{+o} \\ \text{PSP} : \textbf{+en} \end{bmatrix} \Big/ \begin{array}{l} \{\langle \text{PST}, \{\langle \textbf{+o}, \textbf{stg-vb}\rangle, \langle \textbf{+ed}, \textbf{verb}\rangle\}\rangle, \\ \langle \text{PSP}, \{\langle \textbf{+o}, \textbf{stg-vb}\rangle, \langle \textbf{+ed}, \textbf{verb}\rangle\}\rangle\} \end{array}$$

Figure 9: Order Independence Through Tails

We want $\stackrel{<>}{\sqcap}$ to reduce to typed unification when there's no conflict between the FSs to be unified. So $1 \stackrel{<>}{\sqcap} 2$ must be as follows:

$$1 \stackrel{<>}{\sqcap} 2 = \begin{bmatrix} \mathbf{t}_1^1 \\ \text{F} : /\mathbf{a}^1 \end{bmatrix}$$

And $(1 \stackrel{<>}{\sqcap} 2) \stackrel{<>}{\sqcap} 3$ is determined by the fact that indices must determine which default survives when there is conflict:

$$(1 \stackrel{<>}{\sqcap} 2) \stackrel{<>}{\sqcap} 3 = \begin{bmatrix} \mathbf{t}_3^3 \\ \text{F} : /\mathbf{d}^3 \end{bmatrix}$$

Similarly, $2 \stackrel{<>}{\sqcap} 3$ must be as below, if $\stackrel{<>}{\sqcap}$ reduces to monotonic typed unification when there's no conflict:

$$2 \stackrel{<>}{\sqcap} 3 = \begin{bmatrix} \mathbf{t}_3^3 \\ \text{F} : /\mathbf{c}^3 \end{bmatrix}$$

And so $(2 \stackrel{<>}{\sqcap} 3) \stackrel{<>}{\sqcap} 1$ is again determined by the specificity of the indices, since there is conflict:

$$(2 \stackrel{<>}{\sqcap} 3) \stackrel{<>}{\sqcap} 1 = \begin{bmatrix} \mathbf{t}_3^3 \\ \text{F} : /\mathbf{c}^3 \end{bmatrix}$$

This is different from $(1 \stackrel{<>}{\sqcap} 2) \stackrel{<>}{\sqcap} 3$.

The problem is that sometimes the specificity of the default path determines the result in unification: specifically, when two defaults conflict, and one originated from a more specific TDFS than the other. On the other hand, because we want default unification to reduce to monotonic typed unification when there is no conflict, the values on the paths can also sometimes determine the results of persistent default unification: specifically, when there's no conflict. The above example shows that these two factors don't work independently, and therefore unless the dependence between them is recorded in some way, order dependence will result.

This example demonstrates that the indices don't record enough information from the unification history. We must also keep track of values that have been 'overridden', because they must sometimes influence the result of subsequent unifications. This is achieved by tails, as shown briefly below, and in more detail in §5.4. According to the above definition of $\overset{<>}{\sqcap}$, the results are as follows:

$$1 \overset{<>}{\sqcap} 2 = \begin{bmatrix} \mathbf{t}_1 \\ \mathrm{F} : /\mathbf{a} \end{bmatrix} / \{\langle F, \{\langle a, t_1 \rangle, \langle b, t_2 \rangle\}\rangle\}$$

$$(1 \overset{<>}{\sqcap} 2) \overset{<>}{\sqcap} 3 = \begin{bmatrix} \mathbf{t}_3 \\ \mathrm{F} : /\mathbf{c} \end{bmatrix} / \{\langle F, \{\langle a, t_1 \rangle, \langle b, t_2 \rangle, \langle d, t_3 \rangle\}\rangle\}$$

$$2 \overset{<>}{\sqcap} 3 = \begin{bmatrix} \mathbf{t}_3 \\ \mathrm{F} : /\mathbf{c} \end{bmatrix} / \{\langle F, \{\langle b, t_2 \rangle, \langle d, t_3 \rangle\}\rangle\}$$

$$(2 \overset{<>}{\sqcap} 3) \overset{<>}{\sqcap} 1 = \begin{bmatrix} \mathbf{t}_3 \\ \mathrm{F} : /\mathbf{c} \end{bmatrix} / \{\langle F, \{\langle b, t_2 \rangle, \langle d, t_3 \rangle, \langle a, t_1 \rangle\}\rangle\}$$

In this informal presentation of persistent default unification, we have not proved properties such as order independence. This is done in §5. There are two limitations to the operation as we have defined it. Firstly, specificity is determined by an inheritance hierarchy which allows only *strict* inheritance links between the types rather than defeasible ones. We leave open how we could extend the prioritisation of defaults to cases where the priorities themselves were default statements. Secondly, although any node in the FS may be typed, the operation does not take account of any indefeasible type constraints which may apply and thus the result may not be well-formed. We discuss this issue in detail in §4.4, and Lascarides and Copestake (1995) extend the operation defined here, to overcome this limitation.

4 Exploiting Persistent Default Unification

In this section, we briefly cover a number of topics concerning the implementation of persistent associative default unification and its integration in a representation language. We first describe an algorithm for computing $\overset{<>}{\sqcap}$ and show that it is of polynomial complexity, then discuss inheritance hierarchies and how TDFSs can be converted into TFSs. At this point, we can compare persistent default unification to asymmetric default unification and show the latter corresponds to a special case of persistent default unification combined with the conversion of persistent default to non-default FSs. Finally, we make some remarks on well-formedness with respect to a type system.

4.1 Implementation and complexity

The algorithm for computing the results of persistent default unification ($\overset{<>}{\sqcap}$) can be described in terms of the informal definition given in §3.2. It involves the following steps:

1. Construct an indefeasible FS by monotonic unification.

2. Construct a FS for the defeasible reentrancy by monotonic unification of the reentrancy information in the defeasible parts of the input FSs.

3. Merge the tails.

4. Calculate the defeasible FS values.

By examining the informal definition of $\overset{<>}{\sqcap}$ (and the conditions on the logical axioms in §5) it is obvious that these steps can be carried out in this order because no step depends on the results of any later one. The first two steps have the same order of complexity as monotonic unification which is near linear in the number of nodes in the input FSs (e.g. Carpenter, 1992).[5]

The third step, the calculation of tails for each path π, involves constructing the union of the value/specificity pairs for each tail for each path π' which is equivalent to π due to reentrancy.

$$\pi^{12} = \{\pi'^{1} : \langle\pi\rangle \approx_{D_{12}} \langle\pi'\rangle\} \cup \{\pi'^{2} : \langle\pi\rangle \approx_{D_{12}} \langle\pi'\rangle\}$$

At first sight it appears that the resulting tail could contain more value/specificity pairs than were in the originals. However, information about reentrant paths can be amalgamated, because tails for reentrant paths will be equivalent and reentrancy accumulates monotonically in the default FS. In fact, we can modify the notation, so that the first member of each element in a tail is a set of reentrant paths rather than a single path, for example, instead of (6) we can write (7) if F, G and H are reentrant:

(6) $\{\langle F, \{\langle a,t\rangle\ \langle c,t\rangle\}\rangle, \langle G, \{\langle a,t\rangle, \langle c,t\rangle\}\rangle \langle H, \{\langle a,t\rangle, \langle c,t\rangle\}\rangle\}$

(7) $\{\langle\{F, G, H\}, \{\langle a,t\rangle\ \langle c,t\rangle\}\rangle\}$

In this case, the maximum number of tail elements (i.e. path set/value set pairs) is equal to the number of nodes in the default result FS. Merging tails involves using set union to construct each such element, where the maximal number of value/specificity pairs in each value set will be the number of TDFSs in the original KB which are being unified. In practice, tails will normally be much smaller than this implies, because we only keep track of default values which are more specific than the non-default values in the original KB. Furthermore, although for simplicity we described tails so that they can contain values which are inconsistent with the non-default values in the TDFS, such values never have an impact on the unification by the definitions of *Spec* and *Comp*. So in practice, they could be removed from the tail.

The final step is the calculation of the default FS values. For convenience, we repeat the earlier definition of the three values to be unified:

Indef The indefeasible value:
$Indef^{12}_{\pi}$ is the unification of the indefeasible values in I_{12} on all the paths that are reentrant with π in D_{12}

$Indef^{12}_{\pi} = \sqcap\{\phi_{\pi'} : \langle\pi'\rangle =_{I_{12}} \phi_{\pi'} \wedge \langle\pi\rangle \approx_{D_{12}} \langle\pi'\rangle\}$

Spec The most specific default values:
$Spec^{12}_{\pi} = \{\phi : \langle\phi, t_\phi\rangle \in \pi^{12} \text{ and } \forall\langle\phi', t_{\phi'}\rangle \in \pi^{12}, t_{\phi'} \not\sqsubset t_\phi \text{ and } \phi \sqcap Indef^{12}_{\pi} \neq \bot\}$

Comp The compatible default values:
$Comp^{12}_{\pi} = \{\phi : \langle\phi, t_\phi\rangle \in \pi^{12} \text{ and } \forall v \in Spec^{12}_{\pi}, v \sqcap \phi \neq \bot \text{ and } \phi \sqcap Indef^{12}_{\pi} \neq \bot\}$

[5] Assuming types on nodes, requires an extra step in the algorithm over the untyped case to find the greatest common subtype, but since we are assuming that this is unique, this just adds a lookup overhead which will depend on the number of types. For the moment we ignore the extra complexity which arises for well-typed unification, see §4.4, below.

Note that the defeasible value of a path can be determined without reference to the defeasible values on other nodes or to previous defeasible values on any node: the result is just dependent on the indefeasible structure, the reentrancy and the tail. Thus the result can be calculated individually for each node and subsequent calculations of defeasible values on different paths will not alter the result. The worst case complexity is where there is maximal reentrancy, since this gives the longest tails.[6]

We will derive the complexity for a node in a default FS with p paths leading to it, after k FSs from the original KB have been considered, which gives a maximum tail length of $t = kp$:

1. *Spec* is the set of the maximally specific values in the tails, excluding those which are incompatible with the non-default FS. Constructing this set involves $O(t^2)$ comparisons for specificity and t type unification checks with the non-default values in (a).

2. *Comp* is the set of all values in the tails which unify with every member of *Spec* and with the non-default value. This involves $O(t^2)$ type checks.

3. The final step is to unify together all the members of (b) and (c) with the non-default values. This is $O(t)$ in the number of type unifications.

Thus, if we have n nodes in the resulting FS, the overall complexity of the calculation of the default values is $O(nt^2)$. So, although the complexity of $\overset{<>}{\sqcap}$ is worse than that of monotonic typed unification, none of the steps for each node in the result involves complexity worse than $O(t^2)$ in the length of the tails, and these will normally be short.

It is worth noting that it is not actually necessary to calculate default FS values at each step if we know we are applying a sequence of $\overset{<>}{\sqcap}$ operations: the intermediate results are irrelevant, since the values on the default FS can be calculated by looking at the indefeasible FS, the default reentrancy and paths and the tails. (This is formally proved by Lemma 1 in §5.5.) So, for each step apart from the last one within a sequence of $\overset{<>}{\sqcap}$ operations, the only essential overhead compared with monotonic typed unification is calculation of the default reentrancy and merging the tails.

4.2 Inheritance hierarchies

In inheritance hierarchies such as that shown in Figure 7, $\overset{<>}{\sqcap}$ can be used to implement inheritance in a straightforward way. A type t will have a constraint $C(t)$ which is calculated by persistent default unifying its constraint specification with those of its supertypes. Because of order independence, this is equivalent to defining the constraint to be the persistent default unification of the constraint specification with the constraints of the immediate supertypes. For example,

$$
\begin{aligned}
C(\textbf{pst-t-vb}) &= C(\textbf{verb}) \overset{<>}{\sqcap} \begin{bmatrix}\textbf{pst-t-vb}\\ \text{PST} : /\textbf{+t}\end{bmatrix} \\
&= \begin{bmatrix}\textbf{pst-t-vb}\\ \text{PST} : /\boxed{1}\textbf{+t}\\ \text{PSP} : /\boxed{1}\end{bmatrix} \Big/ \begin{array}{l}\{\langle \text{PST}, \{\langle \textbf{+ed}, \textbf{verb}\rangle, \langle \textbf{+t}, \textbf{pst-t-verb}\rangle\}\rangle,\\ \langle \text{PSP}, \{\langle \textbf{+ed}, \textbf{verb}\rangle, \langle \textbf{+t}, \textbf{pst-t-verb}\rangle\}\rangle\}\end{array}
\end{aligned} \tag{8}
$$

[6] If we allow reentrancy to be overridden by default values, as in Lascarides and Copestake (1995), interactions can occur between nodes, and the algorithm to compute this version of default unification has exponential complexity in the worst case. See also §4.4, below, for discussion of imposing well-formedness conditions, in which nodes could interact.

For type hierarchies, we follow the convention that the type of a constraint specification FS is the type being defined and therefore the correct specificity order is built into the operation automatically. But if we were defining inheritance of classes in an untyped framework, all that would be necessary would be to define tails so that instead of types defining the specificity of values we used a tag which reflected the partial order of classes in the inheritance hierarchy. Nothing in our definition of $\overset{<>}{\sqcap}$ apart from this initial instantiation depends on specificity being determined by a type hierarchy. Multiple inheritance in either case needs no special stipulations, although clearly if a type or class has two or more immediate parents these will not be ordered with respect to one another and therefore any conflicts in the default parts of their constraints will lead to nodes with value $\perp$.

Most previous work on default unification has assumed that defaults do not persist outside the lexicon, and that syntactic and semantic specification can be couched entirely in terms of monotonic constraints as is done in current HPSG, for example. Within our current framework this would mean we have to 'convert' persistent defaults which have survived a series of default unifications into indefeasible specifications. For example, the morphological specifications in Figure 7 must not remain default at the point when the resulting FSs are utilised in parsing, otherwise a *+ing* phonological form may unify with an FS default specifying a *+ed* suffix. In many cases of lexical inheritance, default inheritance is for convenience of description, and the ultimate objects of the description are intended to be monotonic TFSs. For **pst-t-verb**, for example, we want to be able to derive the TFS:

$$\begin{bmatrix} \textbf{pst-t-vb} \\ \text{PST} : \boxed{1}\,\textbf{+t} \\ \text{PSP} : \boxed{1} \end{bmatrix}$$

We therefore need to define an operation which combines the defeasible information with the indefeasible structure, which we can apply when we know that all necessary default unification operations have been carried out, for example at the interface between the lexicon and the syntagmatic component. This operation, however, is not an integral part of the definition of persistent default unification, since it is only relevant to its use where defaults are used to describe objects which are treated as ordinary TFSs with respect to some other level of processing. In fact, in §6, we will assume that some classes of defaults persist beyond the lexicon, while others are encapsulated within it. (Even for the latter class, persistence within the lexicon is important, because it allows defaults to be accumulated incrementally, without the sensitivity to textual order in descriptions found in most previous uses of lexical defaults.) For expository convenience we will describe the operation as converting the entire TDFS into a standard TFS, but this easily generalises to the case where only part of the default FS is made indefeasible.

It should be clear from §2 that any operation which makes default information non-default cannot be applied within a series of default unifications without the result being order dependent, so it only makes sense to apply the conversion after all applications of $\overset{<>}{\sqcap}$ within a given 'module', such as the lexicon. We refer to the conversion operation as DefFill, because it fills in the non-default structure from the default information and is thus roughly analogous to the Fill operation which is applied to make a TFS well-formed with respect to a type hierarchy (cf Carpenter, 1992). We first give a definition of DefFill, and then explain why it takes this form.

We define *DefFill*(F), the result of converting a TDFS F into a TFS, as the result of unifying the non-default part of the TDFS with a FS derived from the default part, by ignoring all nodes

$$
\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \begin{bmatrix} \mathrm{H} : \mathbf{a} \end{bmatrix} \\ \mathrm{G} : \begin{bmatrix} \mathrm{H} : \mathbf{b} \end{bmatrix} \end{bmatrix}
\overset{<>}{\sqcap}
\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : /\boxed{1} \\ \mathrm{G} : /\boxed{1} \end{bmatrix}
=
\overset{a \sqcap b = \bot}{\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \begin{bmatrix} \mathrm{H} : \mathbf{a} \end{bmatrix} \\ \mathrm{G} : \begin{bmatrix} \mathrm{H} : \mathbf{b} \end{bmatrix} \end{bmatrix}}
\Big/
\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \boxed{1}\begin{bmatrix} \mathrm{H} : \bot \end{bmatrix} \\ \mathrm{G} : \boxed{1} \end{bmatrix}
$$

Figure 10: Reentrancy Causing Value Conflict

which are typed $\bot$, all nodes which have features and atomic types and all δ-descendants of these nodes. With respect to reentrancy, we only incorporate it into the non-default if there is no inconsistent node either before or after the reentrant node being considered.

Definition 4 *Let $F =_{def} I/D/T$. Then* DefFill(F) *is the most general* TFS *which satisfies the following:*

1. DefFill(F) $\sqsubseteq I$ *i.e. it is subsumed by the indefeasible part of F*
2. *If m_1 the root node of* DefFill(F) *and n_2 is the root node of D then for all* consistent *paths π $\theta(\delta(\pi, m_1)) = \theta(\delta(\pi, n_2))$*
3. *For all consistent paths π and π' such that $\delta(\pi, n_2) = \delta(\pi', n_2)$ then, if all paths $\pi \cdot \pi''$ are consistent, $\delta(\pi, m_1) = \delta(\pi', m_1)$*

Where we define a path π to be consistent *if there are no (possibly empty) paths π' and π'' such that $\pi' \cdot \pi'' = \pi$ and either*

- $\theta(\delta(\pi', n_2)) = \bot$
- $\theta(\delta(\pi', n_2)) = a$ *where a is defined to be an atomic type and $\delta(f, \delta(\pi', n_2))$ is defined for any feature f*

Because of the condition that all values on the nodes in the default FS must be consistent with those in the non-default FS, the TDFS to TFS conversion operation essentially involves adding in the default values to the indefeasible part of the TDFS ignoring nodes typed with $\bot$ and all their δ-descendants (because $\bot$ implies everything, and therefore we cannot assume anything about nodes which follow $\bot$). However, the definition above reflects two complications, the first concerning reentrancy, and the second a (very weak) notion of well-formedness.

The problem with reentrancy is that it is retained in the default FS even if there is a clash of indefeasible values, and therefore we cannot guarantee that reentrancy is compatible with the non-default structure if there is any node typed $\bot$ which is a δ-descendant of the reentrant node. For example, consider Figure 10. We need to discard the reentrancy from the default FS when constructing the TFS, and for this particular case we can therefore add nothing to the non-default structure. Unfortunately, we cannot distinguish between cases such as this where the reentrancy is incompatible because of the indefeasible values and those where reentrancy is compatible with the indefeasible structure, but there has been a clash of default values which is unrelated to the reentrancy. An example of this is shown in Figure 11. The distinction between $\bot$ caused by conflict in indefeasible values and conflict in defeasible values (in this case on the same path), cannot be deduced simply by looking at the tail since a conflict between the indefeasible values c and d might also have been present.

$$a \sqcap b = \bot$$

$$\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \begin{bmatrix} \mathrm{H} : \mathbf{c/a} \end{bmatrix} \\ \mathrm{G} : \begin{bmatrix} \mathrm{H} : \mathbf{d} \\ \mathrm{I} : \mathbf{e} \end{bmatrix} \end{bmatrix} \stackrel{<>}{\sqcap} \begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \begin{bmatrix} \mathrm{H} : \top \end{bmatrix} \\ \mathrm{G} : \begin{bmatrix} \mathrm{H} : \top \end{bmatrix} \end{bmatrix} \Big/ \begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \boxed{1}\begin{bmatrix} \mathrm{H} : \mathbf{b} \end{bmatrix} \\ \mathrm{G} : \boxed{1} \end{bmatrix} =$$

$$\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \begin{bmatrix} \mathrm{H} : \mathbf{c} \end{bmatrix} \\ \mathrm{G} : \begin{bmatrix} \mathrm{H} : \mathbf{d} \\ \mathrm{I} : \mathbf{e} \end{bmatrix} \end{bmatrix} \Big/ \begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \boxed{1}\begin{bmatrix} \mathrm{H} : \bot \\ \mathrm{I} : \mathbf{e} \end{bmatrix} \\ \mathrm{G} : \boxed{1} \end{bmatrix} \Big/ \begin{matrix} \{\langle F \cdot H, \{\langle b,t\rangle, \langle a,t\rangle\}\rangle, \\ \langle G \cdot H, \{\langle b,t\rangle, \langle a,t\rangle\}\rangle\} \end{matrix}$$

Figure 11: Values Conflicting Independently of Reentrancy

$$\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \begin{bmatrix} \mathrm{H} : \mathbf{c} \\ \mathrm{I} : \boxed{1}\ \mathbf{e} \end{bmatrix} \\ \mathrm{G} : \begin{bmatrix} \mathrm{H} : \mathbf{d} \\ \mathrm{I} : \boxed{1} \end{bmatrix} \end{bmatrix}$$

Figure 12: Loss of Reentrancy when Nodes are typed $\bot$

We have adopted the approach of ignoring default reentrancy on a node which is typed $\bot$ or has $\bot$ as a successor, even though this implies that information which could have been incorporated from the default is neglected in some cases. The only other option would be for the *DefFill* operation to itself carry out a check for compatibility with the non-default. We currently think this is less attractive, both theoretically, because this is making *DefFill* do work which should properly be part of the logic of persistent default unification, and practically, because it would need to check all possible combinations of default reentrancy with the non-default FS, an operation which is potentially of exponential complexity. The result of converting the TDFS in Figure 11 to a TFS is shown in Figure 12 (this will be obtained irrespective of the compatibility of c and d). The reentrancy between paths F$\cdot$ I and G$\cdot$I is retained because the corresponding node in the default FS had no δ-descendants marked with $\bot$, but the reentrancy between F and G and between F$\cdot$H and G$\cdot$H is lost because F$\cdot$H/G$\cdot$H was marked $\bot$.

The well-formedness problem is that $\stackrel{<>}{\sqcap}$ will not in general yield a TDFS with a default component which is well-formed, even ignoring $\bot$ nodes. The result of $\stackrel{<>}{\sqcap}$ does not necessarily conform to even the weakest notion of well-formedness: that there are some atomic types for which no features are appropriate. (We will discuss stronger notions of well-formedness with respect to a type system in §4.4.) Thus, as things stand, our system is too weak to be able to define an analogue of untyped FS systems, where nodes which have (atomic) values cannot also have features. For example, if $t \sqsubset t'$, then

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

even though for some types a we want to be able to exclude as not well formed FSs such as:

$$\begin{bmatrix} \mathbf{t} \\ \text{F} : \begin{bmatrix} \mathbf{a} \\ \text{G} : \top \end{bmatrix} \end{bmatrix}$$

One possibility, of course, would be to attempt to define $\overset{<>}{\sqcap}$ so that structures like this were excluded. But it is not clear that this is possible without order dependence. For example, suppose we instead defined $\overset{<>}{\sqcap}$ so that the result of the example above was the following:

$$\begin{bmatrix} \mathbf{t} \\ \text{F} : /\mathbf{a} \end{bmatrix}$$

It is then not clear what to do about cases where there is reentrancy, such as:

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

The reentrancy cannot simply be lost, because then further default unification with a FS with a more specific type than **t** which 'reinstated' the path F·G would result in order dependence. Furthermore, although it may intuitively look reasonable to drop the feature G if the atomic type is preferred, what we have here is not a conflict between values but a conflict between a value and a path. It seems preferable to regard the imposition of well-formedness conditions as part of the operation of incorporation of default information into the non-default FS. It is, after all, clear that the default part of a TDFS is potentially always ill-formed as a FS since it admits nodes typed $\bot$, so it seems unreasonable to impose any stricter well-formedness conditions.

With respect to the very weak notion of well-formedness described so far, the resulting TFS will be well-formed. The notion of typing normally implies stricter well-formedness conditions and we consider some of these in §4.4 below. But first we briefly examine how our definition of persistent default unification relates to previous definitions of default unification as priority union.

4.3 Persistent default unification and priority union

The asymmetric or priority union class of definitions of default unification can now be seen as very special cases of $\overset{<>}{\sqcap}$ combined with the default incorporation function defined above. Apart from Copestake (1993) and Grover *et al* (1994), all definitions of priority union of which we are aware have involved untyped feature structures. This is equivalent in the current formulation to having a type lattice which is completely flat apart from $\top$ and $\bot$, and consists of a series of atomic types corresponding to values and a single non-atomic type, t. The priority union, $\overset{<}{\sqcap}$ of two FS $X \overset{<}{\sqcap} Y$ corresponds to $\mathit{DefFill}(X' \overset{<>}{\sqcap} Y')$ where $X' = X'/X'/T_X$ and $Y' = \top/Y'/T_Y$ (i.e. X' contains only indefeasible information and Y' only defeasible information) and X' and Y' both have type t (so there is no specificity order). Of the definitions surveyed in Copestake (1993), the closest to the behaviour of $\overset{<>}{\sqcap}$ in this special case is Carpenter's skeptical version of default unification. One case where Carpenter's definition does give different results is shown in Figures 13 and 14.

$$
\begin{bmatrix} \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{a} \\ \mathrm{H} : \mathbf{b} \end{bmatrix} \\ \mathrm{I} : \top \end{bmatrix}
\overset{<}{\sqcap}
\begin{bmatrix} \mathrm{F} : \boxed{1} \begin{bmatrix} \mathrm{G} : \boxed{2} \\ \mathrm{H} : \boxed{2} \end{bmatrix} \\ \mathrm{I} : \boxed{1} \end{bmatrix}
=
\begin{bmatrix} \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{a} \\ \mathrm{H} : \mathbf{b} \end{bmatrix} \\ \mathrm{I} : \begin{bmatrix} \mathrm{G} : \top \\ \mathrm{H} : \top \end{bmatrix} \end{bmatrix}
$$

Figure 13: Default unification by Carpenter's definition.

$$
\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{a} \\ \mathrm{H} : \mathbf{b} \end{bmatrix} \\ \mathrm{I} : \top \end{bmatrix}
\overset{<>}{\sqcap}
\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : \top \\ \mathrm{H} : \top \end{bmatrix} \\ \mathrm{I} : \top \end{bmatrix}
\Bigg/
\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \boxed{1} \begin{bmatrix} \mathrm{G} : \boxed{2} \\ \mathrm{H} : \boxed{2} \end{bmatrix} \\ \mathrm{I} : \boxed{1} \end{bmatrix}
=
$$

$$
\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{a} \\ \mathrm{H} : \mathbf{b} \end{bmatrix} \\ \mathrm{I} : \top \end{bmatrix}
\Bigg/
\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \boxed{1} \begin{bmatrix} \mathrm{G} : \boxed{2}\bot \\ \mathrm{H} : \boxed{2} \end{bmatrix} \\ \mathrm{I} : \boxed{1} \end{bmatrix}
$$

$$
\mathit{DefFill}(
\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{a} \\ \mathrm{H} : \mathbf{b} \end{bmatrix} \\ \mathrm{I} : \top \end{bmatrix}
\Bigg/
\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \boxed{1} \begin{bmatrix} \mathrm{G} : \boxed{2}\bot \\ \mathrm{H} : \boxed{2} \end{bmatrix} \\ \mathrm{I} : \boxed{1} \end{bmatrix}
) =
\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{a} \\ \mathrm{H} : \mathbf{b} \end{bmatrix} \\ \mathrm{I} : \top \end{bmatrix}
$$

Figure 14: Persistent Default Unification followed by *DefFill*

The disparity in results is a consequence of the treatment of reentrancy in $\overset{<>}{\sqcap}$ and in *DefFill*, since skeptical default unification retains the maximal information possible, whereas our definition sometimes results in potentially viable information being lost. The advantage of our current version, even considered purely as an implementation of priority union, is that it is computationally tractable, whereas computing the results using Carpenter's definition potentially involves checking all possible combinations of default information for consistency.

4.4 Well-formedness

A variety of different notions of constraints in typed feature structure systems have been proposed, for example by Carpenter (1992), Zajac (1993), Copestake *et al.*(1993) and Gerdemann and King (1994). The ramifications of fully integrating default unification with any one of these systems are considerable. Rather than discussing a particular proposal in detail here, we introduce some of the general issues which have to be considered and outline an approach, which while somewhat restrictive, is adequate for the examples described in this paper.

We first consider well-formedness of TDFSs as a whole. Type hierarchies with constraints differ fundamentally from untyped inheritance systems such as templates in that type constraints on FSs are generally assumed to apply uniformly, however the typed FS arises, whereas templates only apply during a preliminary processing phase associated with the evaluation of FS descriptions. If we consider a description of a template as inheriting from two template FSs the result will be subsumed by their unification, just as a subtype of two types will have a constraint which is subsumed by the unification of the constraints on its parents. But while templates are restricted to the descriptive metalanguage, types are a part of the object

```
Intrans-verb
< SYN CAT > = verb
< SYN SUBJ SYN CAT > = np

Monadic
< SEM PRED > = pred
< SYN SUBJ SEM > = < SEM PRED ARG1 >

Monadic-stative-intrans-verb
Monadic Intrans-verb
< SEM PRED > = stative
```

$$\textbf{intrans-verb} \rightarrow \left[\begin{array}{l} \mathrm{SYN} : \left[\begin{array}{l} \mathrm{CAT} : \textbf{verb} \\ \mathrm{SUBJ} : \left[\mathrm{SYN} : \left[\mathrm{CAT} : \textbf{np} \right] \right] \end{array}\right] \end{array}\right]$$

$$\textbf{monadic} \rightarrow \left[\begin{array}{l} \mathrm{SYN} : \left[\mathrm{SUBJ} : \left[\mathrm{SEM} : \boxed{1} \right] \right] \\ \mathrm{SEM} : \left[\begin{array}{l} \mathrm{PRED} : \textbf{pred} \\ \mathrm{ARG1} : \boxed{1} \end{array}\right] \end{array}\right]$$

$$\textbf{monadic-stative-intrans-verb} \rightarrow \textbf{monadic} \wedge \textbf{intrans-verb} \wedge \left[\mathrm{SEM} : \left[\mathrm{PRED} : \textbf{stative} \right] \right]$$

Figure 15: Templates and types

language. Thus, if we define two templates, Intrans-verb and Monadic as shown in Figure 15 and define Monadic-stative-intrans-verb to inherit from both of them, this does not affect the result of unification of two FSs which happen to be equal to the template descriptions. In contrast, consider the type system also shown in Figure 15 which implies that any time two FSs of types **intrans-verb** and **monadic** are unified, the result must be of type **monadic-stative-intrans-verb** and therefore, to be well-formed, must be subsumed by its constraint, which can be more specific than the unification of its parent constraints. So typed unification defined simply as a version of untyped unification augmented with the calculation of greatest common subtypes for each pair of input nodes does not return a result which respects all type constraints. In general, an extra step may be needed for each node, of unifying in the constraint on that node's type. Well-typed unification, $\sqcap_t$ can be defined in these terms (see Carpenter, 1992:92).

The approaches to typing FSs mentioned above assume that every node in a FS is typed, and that monotonic constraints can be stipulated with respect to any of these types. In this paper, we have limited the notion of a default constraint so that the root node of the constraint FS is typed but we do not allow embedded nodes in constraints to carry types which independently have their own default constraints. Since constraint FSs can be of arbitrary size, this does not prevent any subpart of a FS being default, but it does avoid the complex interactions between different sources of defaults which would otherwise be possible. We also assume that the root type of the default part of the TDFS is equal to the root type of the non-default. Under these conditions, calculation of inheritance as described in §4.2 will give a TDFS which is well-formed in that it is subsumed by the indefeasible part of the constraint on its type, and incorporates all possible information from the defeasible part, but this is only true because we know the type of the result beforehand and are already unifying in its constraint.[7] To ensure persistent default unification always returns a result which is well-formed by this definition, an extra default unification operation may be required, just as it is in the monotonic typed unification case. We thus define a well-formed default unification operation $\overset{<>}{\sqcap}_t$:

Definition 5 $FS_1 \overset{<>}{\sqcap}_t FS_2 = (FS_1 \overset{<>}{\sqcap} FS_2) \overset{<>}{\sqcap} \mathrm{C}(\theta(n_1))$ *where* n_1 *is the root node of the indefeasible part of* $FS_1 \overset{<>}{\sqcap} FS_2$ *and* $\mathrm{C}(t)$ *gives the* TDFS *representing the constraint on* t.

Clearly, since this is just another $\overset{<>}{\sqcap}$ it is irrelevant at what stage it is carried out within in a series of default unifications. This is not completely analogous to the $\sqcap_t$ operation described above, because there is no need to consider nodes internal to the root node and recursively apply their constraints.

We return now to the question of typing with respect to a series of monotonic constraints on nodes other than the root. The concept of an atomic type given in §4.2 is an example of an appropriateness condition which restricts the range of features possible as transitions from a node of a given type. This is a very weak example of an appropriateness condition: an atomic type has no appropriate features, but all features are possible for all non-atomic types. In general, however, we want to be able to state stricter conditions than this, which will apply in addition to the subsumption condition mentioned above, and which will affect all typed nodes, not just root nodes. The most straightforward case is where we can stipulate the appropriate features for a type, and for each feature independent of type we can stipulate an appropriate value. This can be implemented as part of the incorporation operation in a manner analogous to the atomic type stipulation discussed in §4.2. If we find a node which has a feature which is not appropriate for its type (or any of its subtypes) we ignore that node when constructing the TFS.

This notion of typing is adequate for the examples discussed in this paper. However it is a weaker version of the appropriateness conditions discussed by Carpenter (1992), because we do not allow values to be stipulated by types. Difficulties arise once we allow types to affect values on successor nodes, because we lose independence between node values. Consider the following appropriateness conditions:

$$\begin{array}{l} \mathbf{t3} \rightarrow \left[\mathrm{G} : \mathbf{t2} \right] \\ \mathbf{t4} \rightarrow \left[\mathrm{G} : \mathbf{t5} \right] \end{array}$$

(where $\mathbf{t4} \sqsubset \mathbf{t3}$ and thus $\mathbf{t5} \sqsubset \mathbf{t2}$) and suppose that we have to convert the following TDFS to a TFS:

$$\begin{bmatrix} \mathbf{t1} \\ \mathrm{F} : \begin{bmatrix} \mathbf{t3/t4} \\ \mathrm{G} : \mathbf{t6/t7} \end{bmatrix} \end{bmatrix}$$

[7] In the monotonic case, well-formedness is usually described in terms of subsumption: a constraint on a type will subsume any well-formed FS which has a root node of that type. Obviously we cannot define well-formedness of TDFSs by subsumption, since constraints may have default components. However, the definition in the monotonic case could be rephrased as a condition that a well-formed FS is unchanged when it is unified with the relevant constraint, and an equivalent definition applies here: any well-formed TDFS will be unchanged by the result of persistent default unification with the constraint on its root node type.

If the appropriateness conditions on **t3** or **t4** induce a conflict with the type on F·G the result is potentially indeterminate. For example, assume **t2** is compatible with both **t6** and **t7**, **t5** is compatible with **t6** but $\mathbf{t5} \sqcap \mathbf{t7} = \bot$. In this case, the TFS which results from incorporating both defaults is not well-formed (9), but there multiple well-formed possibilities according to whether we accept **t4** in preference to **t7** (10), **t7** instead of **t4** (11) or neither default type (12).

(9) $$\begin{bmatrix} \mathbf{t1} \\ \mathrm{F} : \begin{bmatrix} \mathbf{t4} \\ \mathrm{G} : \mathbf{t7} \end{bmatrix} \end{bmatrix}$$

(10) $$\begin{bmatrix} \mathbf{t1} \\ \mathrm{F} : \begin{bmatrix} \mathbf{t4} \\ \mathrm{G} : \mathbf{t6} \end{bmatrix} \end{bmatrix}$$

(11) $$\begin{bmatrix} \mathbf{t1} \\ \mathrm{F} : \begin{bmatrix} \mathbf{t3} \\ \mathrm{G} : \mathbf{t7} \end{bmatrix} \end{bmatrix}$$

(12) $$\begin{bmatrix} \mathbf{t1} \\ \mathrm{F} : \begin{bmatrix} \mathbf{t3} \\ \mathrm{G} : \mathbf{t6} \end{bmatrix} \end{bmatrix}$$

The result most in accord with our previous assumptions would be (12), but this means that *DefFill* cannot be computed without considering all the interactions of appropriateness conditions, which becomes particularly complex when we consider examples with reentrancy. Thus, the interaction of constraints and defaults is not straightforward, but we will not attempt to resolve the problem here, since the options depend on the precise assumptions made about typing, and are essentially independent of the $\overset{<>}{\sqcap}$ operation itself, under the assumption that this is not required to return a well-formed FS. The only definition of typed default unification of which we are aware which does explicitly address the issue of interaction of default and type constraints is that in Copestake (1993). This treats default unification as an operation to be constrained by a type system, not integrated with it, and is in any case somewhat inelegant. We therefore leave open here the issue of how to integrate our definition of persistent default unification with a full type based constraint system. However, Lascarides and Copestake (1995) propose an extension to persistent default unification, which can be integrated in a straightforward manner with such a system.

5 The Logic of Persistent Default Unification

The rest of the paper can be understood without reading this section. However, specifying the logic which defines the operation $\overset{<>}{\sqcap}$ allows us to prove that it meets the criteria we specified in section 3.2. We will translate TDFSs into a modal language with well-defined syntax and semantics; define the axioms of the logic, and the accompanying logical consequence relation; define the operation $\overset{<>}{\sqcap}$ in terms of this logical consequence relation; and then prove that $\overset{<>}{\sqcap}$ has the properties specified in section 3.2.

5.1 Translating TDFSs into a Modal Language

We will use a conditional logic to axiomatise $\overset{<>}{\sqcap}$. We use a modal language to describe the paths in the FSs (cf. Blackburn, 1992). We introduce four conditionals into the language, which will be used to represent: (i) indefeasible paths with values; (ii) indefeasible reentrancies; (iii) defeasible paths with values; and (iv) defeasible reentrancies. Indefeasible paths will be translated as: $1 \rightsquigarrow \langle\pi\rangle a$, where 1 is the tag of the TDFS, π is a path, or sequence of features, and a is a type. Semantically this means that the TDFS 1 indefeasibly entails the path $\langle\pi\rangle a$, and for any path $\langle\pi\rangle b$ indefeasibly entailed by this TDFS, $a \sqsubseteq b$. So in a sense, $\langle\pi\rangle a$ is the $\langle\pi\rangle$-prime implicate. Non-default reentrancies will be translated as: $1 \rightarrow \langle\pi\rangle \approx \langle\pi'\rangle$, where $\rightarrow$ is the material conditional, and $\approx$ is the Kasper-Rounds connective for defining reentrancy (Kasper and Rounds, 1986). Default paths will be translated as $1 \Rightarrow \langle\pi\rangle a$. Semantically, this means that the TDFS 1 defeasibly entails $\langle\pi\rangle a$, and if it defeasibly entails $\langle\pi\rangle b$, then $a \sqsubseteq b$. Finally, default reentrancies are translated as $1 > \langle\pi\rangle \approx \langle\pi'\rangle$. Semantically, this means that the TDFS 1 defeasibly entails the reentrancy between paths $\langle\pi\rangle$ and $\langle\pi'\rangle$. Note that this doesn't preclude other reentrancies involving these paths also being entailed by the TDFS.

We first define the syntax and semantics of the modal language $\mathcal{L}_{pdu}$, in which $\overset{<>}{\sqcap}$ will be modelled.

The Syntax of $\mathcal{L}_{pdu}$

1. Types $t, t', a, b, \top, \bot \ldots$ are WFFs.
   (Types in a TDFS will be translated into these WFFs).

2. Indices $1, 2, \ldots$ are WFFs.
   (The index in a TDFS will be translated into these WFFs).

3. If 1 is an index, then $\hat{1}$ is a term.[8]

4. If π is a path of features, and a is a type, then $\langle\pi\rangle a$ and $[\pi]a$ are WFFs.

5. If π and π' are paths, then $\langle\pi\rangle \approx \langle\pi'\rangle$ is a WFF.

6. If ϕ and ψ are WFFs, then so are , $\phi \wedge \psi$, $\phi \rightarrow \psi$ and $\phi > \psi$.

7. If 1 is a tag and $\langle\pi\rangle a$ is a path, then $1 \Rightarrow \langle\pi\rangle a$ and $1 \rightsquigarrow \langle\pi\rangle a$ are WFFS.

We will also occasionally use $\sqcap$ in the syntax of the language, for notational convenience. $a \sqcap b$ will be shorthand for that value c which is the value of $a \sqcap b$ according to the type hierarchy $\langle \mathsf{Type}, \sqsubseteq \rangle$.

A further notational convention we will use is: $A \not\rightsquigarrow B$ for $(A \rightsquigarrow B) \rightarrow \bot$.[9] Similarly for $A \not\rightarrow B$, $A \not\Rightarrow B$, and $A \not> B$.

We also use $\sqcap$ to refer to the infinum of sets of types. For example, if $X = \{a, b, c\}$, then we will write $\sqcap X$ for $a \sqcap b \sqcap c$. And if $Y = \{d, e\}$, then we will write $X \sqcap Y$ for $a \sqcap b \sqcap c \sqcap d \sqcap e$.

[8]The term $\hat{1}$ will correspond to the tag on a TDFS.

[9]Strictly speaking, the languge contains two $\bot$s; one which stands for logical inconsistency—which is the one used in $A \not\rightsquigarrow B$—and the other which stands for the most specific type in $\langle \mathsf{Type}, \sqsubseteq \rangle$.

The Semantics of $\mathcal{L}_{pdu}$

A model for the language $\mathcal{L}_{pdu}$ is a tuple $\langle W, *, R_F, R_G, \ldots, I\rangle$, where:

- W is a (non-empty) set of nodes;
- $*$ is a function from $W \times \mathcal{P}(W)$ to $\mathcal{P}(W)$, where $\mathcal{P}(W)$ is the power set of W.
  The constraints on $*$ will be determined by the axioms of the logic that we define shortly. These axioms will determine how defaults behave during unification.
- $R_F, R_G, \ldots$ are accessibility relations on W (one for each feature in Feat).
  The constraints on these relations will be determined by the axioms of the logic that we define shortly. They ensure that TDFSs are double DAGs, for example.
- I is the interpretation function, which assigns each type in Type and each tag in N a member of $\mathcal{P}(W)$.

If π is a sequence of features $F_1 \ldots F_n$, then $\langle\pi\rangle = \langle F_1\rangle \ldots \langle F_n\rangle$, and $R_\pi = R_{F_1} \circ \ldots \circ R_{F_n}$

We now define the semantics of the WFFs of the language.

1. Where ϕ is a type or tag $[\![\phi]\!]_M(n) = \mathit{true}$ iff $n \in I(\phi)$.
2. For any feature F, $[\![\langle F\rangle\phi]\!]_M(n) = \mathit{true}$ iff
   there is an n' such that nR_Fn' and $[\![\phi]\!]_M(n') = \mathit{true}$.
3. For any paths π and π', $[\![\langle\pi\rangle \approx \langle\pi'\rangle]\!]_M(n) = \mathit{true}$ iff $n'R_\pi n$ iff $n'R_{\pi'}n$, and
   there exists an n' such that $n'R_\pi n$.
4. $[\![\phi > \psi]\!]_M(n) = \mathit{true}$ just in case $*(n, [\![\phi]\!]_M) \subseteq [\![\psi]\!]_M$
5. $[\![1 \rightsquigarrow \langle\pi\rangle a]\!](n) = \mathit{true}$ just in case $[\![1 \rightarrow \langle\pi\rangle a]\!](n) = \mathit{true}$
   and if $[\![1 \rightarrow \langle\pi\rangle b]\!](n) = \mathit{true}$ then $[\![a]\!] \subseteq [\![b]\!]$.
6. $[\![1 \Rightarrow \langle\pi\rangle a]\!](n) = \mathit{true}$ just in case $[\![1 > \langle\pi\rangle a]\!](n) = \mathit{true}$
   and if $[\![1 > \langle\pi\rangle b]\!](n) = \mathit{true}$ then $[\![a]\!] \subseteq [\![b]\!]$.

The semantics of $\langle\pi\rangle\phi$ correspond exactly to that given in Blackburn (1992), and the semantics of $\langle\pi\rangle \approx \langle\pi'\rangle$ corresponds to that given in Kasper and Rounds (1986). These formulae can be used to define FSs, as explained in Blackburn (1992). FSs are a particular kind of DAG, and formulae of the form $\langle\pi\rangle\phi \wedge \langle\pi'\rangle\psi \wedge \ldots$ describe such DAGs, as long as appropriate constraints are imposed on the accessibility relations and $\approx$. This is because Kripke models are directed graphs. For example, the FS in (13) describes a DAG, which is denoted by the formula $\langle F\rangle a \wedge \langle G\rangle b \wedge \langle F\rangle \approx \langle G\rangle$, under certain assumptions about accessibility (such as $\langle F\rangle a$ entails $[F]a$).

(13) $\begin{bmatrix} \mathrm{F} : \boxed{1}\,\mathbf{a} \\ \mathrm{G} : \boxed{1} \end{bmatrix}$

The necessary axioms for ensuring this representation of FSs is an accurate one are defined below.

5.2 The Translation of TDFSs into $\mathcal{L}_{pdu}$

The translation from TDFSs into $\mathcal{L}_{pdu}$ is defined in terms of two translation functions: one that takes reentrant nodes as arguments, and the other that takes nodes which are assigned a type as arguments. The tails do not undergo a translation procedure. Rather, there will be definitions that explain how the tails are constructed, both in the initial KB and when default unification is performed. The tails will then play a part in defining the axioms on the conditional $\Rightarrow$. But the tails themselves are in the metalanguage of $\mathcal{L}_{pdu}$.

The translation of the FSs is a bijection, and therefore has an inverse. Each TDFS is translated into $\rightarrow$, $\rightsquigarrow$, $>$ and $\Rightarrow$ rules, with the tag in the antecedent. For example:

1. $1 \rightarrow \langle\pi\rangle \approx \langle\pi'\rangle$
2. $1 \rightsquigarrow \langle\pi\rangle a$
3. $1 > \langle\pi\rangle \approx \langle\pi'\rangle$
4. $1 \Rightarrow \langle\pi\rangle a$

When unifying two TDFSs tagged, say, 1 and 2, we translate the two TDFSs into rules of the above form. The axioms of the logic will then generate new $\rightarrow$, $\rightsquigarrow$, $>$ and $\Rightarrow$ rules, with the tag $1 \wedge 2$ in the antecedent (as before, we will sometimes abbreviate the tag $1 \wedge 2$ to 12). Using the inverse translation on these formulae, and the definition of how a tail is transformed by the $\overset{<>}{\sqcap}$ operation, which is given below, we produce the resulting TDFS, $\text{TDFS}_1 \overset{<>}{\sqcap} \text{TDFS}_2$.

The translation on nodes is defined as follows:

Definition 6 *Let $\langle \mathcal{T}, i, Q, \langle r_1, r_2\rangle, \delta, \theta\rangle$ be a* TDFS. *Then the translation of this* TDFS *is defined in terms of two functions τ_1 and τ_2, which are defined as follows:*

- $\tau_1 : Q_1 \longrightarrow \mathcal{L}_{pdu}$, *where*

  $$Q_1 = \{n \in Q : \exists \text{ paths } \pi, \pi' \text{ such that } n = \delta(r_1, \pi) = \delta(r_1, \pi') \text{ or } n = \delta(r_2, \pi) = \delta(r_2, \pi')\}$$

  and

  $$\tau_1(n) = \begin{cases} \{i \rightarrow \langle\pi\rangle \approx \langle\pi'\rangle\} & \text{If } n = \delta(r_1, \pi) = \delta(r_1, \pi') \\ \{i > \langle\pi\rangle \approx \langle\pi'\rangle\} & \text{If } n = \delta(r_2, \pi) = \delta(r_2, \pi') \end{cases}$$

- $\tau_2 : Q_2 \longrightarrow \mathcal{L}_{pdu}$ *where*

  $$Q_2 = \{n \in Q : \theta(n) \text{ is defined.}\}$$

  and

  $$\tau_2(n) = \begin{cases} \{i \rightsquigarrow \langle\pi\rangle\theta(n) : \delta(r_1, \pi) = n\} & \text{If } n \text{ is a } \delta\text{-descendant of } r_1 \\ \{i \Rightarrow \langle\pi\rangle\theta(n) : \delta(r_2, \pi) = n\} & \text{If } n \text{ is a } \delta\text{-descendant of } r_2 \end{cases}$$

5.2.1 An Example of Translation into $\mathcal{L}_{pdu}$

The translation of the TDFS (14) is (14′).

(14)
$$\begin{bmatrix} \mathbf{t}^1 \\ \text{F} : \begin{bmatrix} \text{G} : \mathbf{a}/\boxed{2} \\ /\text{H} : \boxed{2} \end{bmatrix} \\ \text{H} : \mathbf{b}/\mathbf{c} \end{bmatrix} \quad /\{\langle F \cdot H, \{\langle a, t\rangle\}\rangle, \langle H, \{\langle c, t\rangle\}\rangle\}$$

A tuple that describes this TDFS is $\langle \mathcal{T}, 1, Q, (n_1, n_5), \delta, \theta\rangle$, where:

1. $\mathcal{T} = \{\langle F \cdot H, \{\langle a, t\rangle\}\rangle, \langle H, \{\langle c, t\rangle\}\rangle\}$
2. $Q = \{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8\}$
3. $\delta(n_1, F) = n_2$, $\delta(n_2, G) = n_3$, $\delta(n_1, H) = n_4$, $\delta(n_5, F) = n_6$, $\delta(n_6, G) = n_7$, $\delta(n_6, H) = n_7$, $\delta(n_5, H) = n_8$
4. $\theta(n_1) = t$, $\theta(n_3) = a$, $\theta(n_4) = b$, $\theta(n_5) = t$, $\theta(n_7) = a$, $\theta(n_8) = c$

(14′)
$$\begin{aligned}
F \cdot H^1 = {} & \{\langle a, t\rangle\} \\
H^1 = {} & \{\langle c, t\rangle\} \\
1 \rightsquigarrow {} & t \\
1 \rightsquigarrow {} & \langle F\rangle\langle G\rangle a \\
1 \rightsquigarrow {} & \langle H\rangle b \\
1 > {} & \langle F\rangle\langle G\rangle \approx \langle F\rangle\langle H\rangle \\
1 \Rightarrow {} & t \\
1 \Rightarrow {} & \langle F\rangle\langle G\rangle a \\
1 \Rightarrow {} & \langle F\rangle\langle H\rangle a \\
1 \Rightarrow {} & \langle H\rangle c
\end{aligned}$$

5.3 The Axioms of the Logic

We now define the axioms of the logic, so that we can infer new conditionals from the translations of the TDFSs TDFS_1 and TDFS_2 to be unified. The new conditionals, together with the new tail as defined below, will be the translation of the TDFS $\text{TDFS}_1 \overset{<>}{\sqcap} \text{TDFS}_2$. We will later see that one cannot take an arbitrary theory of $\mathcal{L}_{pdu}$, and determine whether it specifies a TDFS; although this won't matter given the way $\overset{<>}{\sqcap}$ is defined. See section 4.4.

The Axioms on Features and Reentrancy

The following axioms are consistent with those given in Blackburn (1992), and Kasper and Rounds (1986):

PN `A Possible path is a Necessary One`
$\vdash \langle F\rangle\phi \rightarrow [F]\phi$

ARE `Adding Reentrancies`
$\vdash (\langle\pi\rangle \approx \langle\pi'\rangle \wedge \langle\pi'\rangle \approx \langle\pi''\rangle) \rightarrow \langle\pi\rangle \approx \langle\pi''\rangle$

RET `Reentrancy and Top`
$\vdash \langle\pi\rangle \approx \langle\pi'\rangle \rightarrow (\langle\pi\rangle\top \wedge \langle\pi'\rangle\top)$
$\vdash \langle\pi\rangle \approx \langle\pi\rangle \leftrightarrow \langle\pi\rangle\top$

REV `Reentrancies and Values`
$\vdash (\langle\pi\rangle\phi \wedge \langle\pi'\rangle\psi \wedge \langle\pi\rangle \approx \langle\pi'\rangle) \rightarrow$
$(\langle\pi\rangle(\phi \sqcap \psi) \wedge \langle\pi'\rangle(\phi \sqcap \psi))$

ACL `Acycles`
$\vdash (\langle\pi_1\pi_2\rangle \approx \langle\pi_2\rangle) \rightarrow \bot$

These ensure that the modal semantics does indeed describe the FSs in the definition of TDFSs in the appropriate way. For example, as long as the accessibility relations are constrained so as to model PN, each DAG in the model contains at most one $\langle\pi\rangle$-arc—just like FS DAGs); and by ARE, reentrancies are transitive. The semantic constraints corresponding to RET and REV ensure the DAG in the Kripke model is isomorphic to one containing node sharing, when the paths are reentrant. And by ACL, cycles are disallowed.

We also assume the axioms on $\rightarrow$, familiar from first order logic.

The Axioms of $\rightsquigarrow$

The axioms on $\rightarrow$ and $\rightsquigarrow$ determine how the indefeasible information in the TDFSs combines under the operation $\overset{<>}{\sqcap}$.

The first axioms PI1 and PI1$'$ ensure that the path in the consequent of a $\rightsquigarrow$ rule is the most specific path. The last three axioms TU1, TU2 and TU3 ensure that, together with PI1 and PI1$'$, $\rightsquigarrow$ corresponds to typed unification.

PI1 `The First Prime Implicate Axiom: Part I`[10]

$$\frac{1 \rightsquigarrow \langle\pi\rangle a}{(1 \rightsquigarrow \langle\pi\rangle b) \rightarrow b = a}$$

PI1$'$ `The First Prime Implicate Axiom: Part II`

$$\frac{\begin{array}{l}1 \rightarrow \langle\pi\rangle a\\ (1 \rightarrow \langle\pi\rangle b) \rightarrow (a \rightarrow b)\end{array}}{1 \rightsquigarrow \langle\pi\rangle a}$$

TU1 `The First TU Axiom`

$$\frac{\begin{array}{l}1 \rightsquigarrow \langle\pi\rangle a\\ (1 \rightarrow \langle\pi\rangle \approx \langle\pi'\rangle) \rightarrow (2 \not\rightarrow \langle\pi'\rangle\top)\end{array}}{(1 \wedge 2) \rightarrow \langle\pi\rangle a}$$

TU2 `The Second TU Axiom`

$$\frac{\begin{array}{l}1 \rightsquigarrow \langle\pi\rangle a\\ 2 \rightsquigarrow \langle\pi'\rangle b\\ 1 \rightarrow \langle\pi\rangle \approx \langle\pi'\rangle\end{array}}{(1 \wedge 2) \rightarrow \langle\pi\rangle(a \sqcap b)}$$

TU3 `The Third TU Axioms`

$$\frac{1 \rightarrow \langle\pi\rangle\bot}{1 \rightarrow \bot} \quad \text{and} \quad \frac{1 \rightarrow \bot}{1 \rightarrow \langle\pi\rangle\bot}$$

TU1 and PI1$'$ ensure that the following holds:

$$\begin{bmatrix}\mathbf{t}_1^1\\ \mathrm{F}:\mathbf{a}\end{bmatrix} \overset{<>}{\sqcap} \begin{bmatrix}\mathbf{t}_2^2\\ \mathrm{G}:\mathbf{b}\end{bmatrix} = \begin{bmatrix}\mathbf{t}_1 \sqcap \mathbf{t}_2^{(1\wedge 2)}\\ \mathrm{F}:\mathbf{a}\\ \mathrm{G}:\mathbf{b}\end{bmatrix}$$

TU2 and PI1$'$ ensure the following holds:

[10] One could replace $b = a$ in PI1 with $a \sqsubseteq b$; the resulting axiom would be equivalent.

$$\begin{bmatrix} \mathbf{t}_1^1 \\ \text{F}: \boxed{1}\mathbf{a} \\ \text{G}: \boxed{1} \end{bmatrix} \overset{<>}{\sqcap} \begin{bmatrix} \mathbf{t}_2^2 \\ \text{G}: \boxed{2}\mathbf{b} \\ \text{H}: \boxed{2} \end{bmatrix} = \begin{bmatrix} \mathbf{t}_1 \sqcap \mathbf{t}_2^{(1\wedge 2)} \\ \text{F}: \boxed{1}\mathbf{a}\sqcap\mathbf{b} \\ \text{G}: \boxed{1} \\ \text{H}: \boxed{1} \end{bmatrix}$$

TU2, TU3 and PI1′ ensure the following holds: where $a \sqcap b = \bot$:

$$\begin{bmatrix} \mathbf{t}^1 \\ \text{F}: \mathbf{a} \end{bmatrix} \overset{<>}{\sqcap} \begin{bmatrix} \mathbf{t}^2 \\ \text{F}: \mathbf{b} \end{bmatrix} = \bot^{(1\wedge 2)}$$

These axioms ensure that step 1 in the informal description of $\overset{<>}{\sqcap}$ holds. That is, the indefeasible part of $FS_1 \overset{<>}{\sqcap} FS_2$ is $I_1 \sqcap I_2$.

We now consider the axioms for $>$, the definition of tails, and the axioms for $\Rightarrow$. These will determine the default aspect of the definition of $\overset{<>}{\sqcap}$.

The Axioms on $>$

Given the semantics of $>$, we have the following two axioms:

CR `Closure on the Right`

$$\frac{A > B \qquad \vdash B \rightarrow C}{A > C}$$

AND `The And Condition`

$$\frac{A > B \qquad A > C}{A > (B \wedge C)}$$

In addition, we add the following, and a corresponding semantic constraint on $*$:[11]

RE1 `The First Reentrancy Axiom`

$$\frac{1 \rightarrow \langle\pi\rangle \approx \langle\pi'\rangle}{1 > \langle\pi\rangle \approx \langle\pi'\rangle}$$

In other words, any reentrancy in the indefeasible DAG has a corresponding reentrancy in the defeasible DAG. This is in line with the isomorphism constraint between the DAGs in the definition of TDFSs.

There is one more axiom involving $>$ and reentrancy, which will tell us which defeasible reentrancies survive unification:

RE2 `The Second Reentrancy Axiom`

$$\frac{1 > \langle\pi\rangle \approx \langle\pi'\rangle}{(1 \wedge 2) > \langle\pi\rangle \approx \langle\pi'\rangle}$$

[11] For the purposes of brevity, we will not define the semantic constraints on $*$ that correspond to the axioms on $>$ and $\Rightarrow$.

RE2 entails that the definition of $\overset{<>}{\sqcap}$ will be one in which default reentrancies always survive, even when there's incompatible indefeasible information (in which case the default value on the reentrant node is $\bot$). That is, it corresponds to step 2 in the informal definition specified in §3.2. If the reentrancies in 1 and 2 produce a cyle, then by RE2, ARE and CR, $(1 \wedge 2) > \bot$. The definition of a PDU deduction below ensures that in this case, the defeasible part of the unified TDFS is $\bot$.

The Definition of Tails

Tails are defined in the following manner:

- Let the TDFS TDFS_1 be in the initial KB; we call such TDFSs *basic* TDFSs. Then for any path π in TDFS_1:

$$\pi^1 = \{\langle a, t\rangle : \text{ where } 1 \Rightarrow t, 1 \Rightarrow \langle\pi\rangle a, 1 \not\rightsquigarrow \langle\pi\rangle a\}$$

  That is, these tails record just the information that is defeasible but *not* indefeasible, together with the root type (specificity).

- The following rule tells us how to construct new tails from old ones, during unification:

$$\pi^{1\wedge 2} = \cup^{\pi'}_{(1\wedge 2)>\langle\pi\rangle\approx\langle\pi'\rangle}\pi'^1 \cup \pi'^2$$

  This corresponds exactly to the instructions in step 3 of the informal definition of $\overset{<>}{\sqcap}$ given earlier.

When unifying TDFS_1 with TDFS_2, the π-part of the new tail is formed by set union on the tails π'^1 and π'^2, for all π' that are reentrant with π. So, in particular, $\pi^1 \cup \pi^2 \subseteq \pi^{12}$ (since π is reentrant with itself). These two definitions ensure that the tail of a TDFS contains all the strictly defeasible values on the *original* TDFSs that were used to build this TDFS, together with the specificity information. We will see that this definition of tails guarantees the following result:

$$\begin{bmatrix} \mathbf{t} \\ \text{F} : /\boxed{1}\,\mathbf{a} \\ \text{G} : /\boxed{1} \\ \text{H} : \mathbf{b} \end{bmatrix} \Big/ \begin{array}{l}\{\langle F, \{\langle a, t\rangle\}\rangle, \\ \langle G, \{\langle a, t\rangle\}\rangle\}\end{array} \overset{<>}{\sqcap} \begin{bmatrix} \mathbf{t} \\ \text{G} : /\boxed{2}\,\mathbf{c} \\ \text{H} : /\boxed{2} \end{bmatrix} \Big/ \begin{array}{l}\{\langle G, \{\langle c, t\rangle\}\rangle, \\ \langle H, \{\langle c, t\rangle\}\rangle\}\end{array}$$

$$= \begin{bmatrix} \mathbf{t} \\ \text{F} : /\boxed{1}\,\mathbf{a} \sqcap \mathbf{b} \sqcap \mathbf{c} \\ \text{G} : /\boxed{1} \\ \text{H} : \mathbf{b}/\boxed{1} \end{bmatrix} \Big/ \begin{array}{l}\{\langle F, \{\langle a, t\rangle\ \langle c, t\rangle\}\rangle, \\ \langle G, \{\langle a, t\rangle, \langle c, t\rangle\}\rangle \\ \langle H, \{\langle a, t\rangle, \langle c, t\rangle\}\rangle\}\end{array}$$

Note that in the above, $\langle H, \{\langle b, t\rangle\}\rangle$ is not in the tail of the LHS TDFS, because the path $H : b$ is indefeasible (as well as defeasible). Because of this, it is not in the tail of the result either. But the values in the F, G and H parts of the tails are expanded in the result, because of the reentrancy, which determines which sets to unify in the resulting tail. If we defined π^{12} to be $\pi^1 \cup \pi^2$, then in the above, $F^{12} \neq H^{12}$. So we would not get a well-defined TDFS in this example of $\overset{<>}{\sqcap}$, because the axioms on $\Rightarrow$, which are defined using the tails, would fail to see that the default values on F, G, and H must agree.

The Axioms on $\Rightarrow$

Having defined tails, we are finally in a position to define the axioms on $\Rightarrow$, which will ultimately determine which default paths in a TDFS survive the $\overset{<>}{\sqcap}$ operation. There are three axioms: the first ensures that the path in a consequent of a $\Rightarrow$-rule is the most specific; the second and third define the $\overset{<>}{\sqcap}$ operation for default paths.

PI2 `The Second Prime Implicate Axiom`

$$\frac{1 \Rightarrow \langle\pi\rangle a}{(1 \Rightarrow \langle\pi\rangle b) \rightarrow b = a}$$

CON `Consistency`

$$\frac{\begin{array}{l} 1 \Rightarrow \langle\pi\rangle a \\ (1 > \langle\pi\rangle \approx \langle\pi'\rangle) \rightarrow (2 \not> \langle\pi'\rangle\top) \\ (1 \wedge 2) \not> \langle\pi_1\pi_2\rangle \approx \langle\pi_2\rangle \end{array}}{(1 \wedge 2) \Rightarrow \langle\pi\rangle a}$$

PDU `The Persistent Default Unification Axiom`

$$\frac{\begin{array}{l} 1 \Rightarrow \langle\pi\rangle a \\ 2 \Rightarrow \langle\pi'\rangle b \\ 1 > \langle\pi\rangle \approx \langle\pi'\rangle \\ (1 \wedge 2) \not> \langle\pi_1\pi_2\rangle \approx \langle\pi_2\rangle \end{array}}{(1 \wedge 2) \Rightarrow \langle\pi\rangle(\mathit{Spec}_\pi^{12} \sqcap \mathit{Comp}_\pi^{12} \sqcap \psi)}$$

Where:

$\psi = \sqcap\{\psi_{\pi_i} : 1 \wedge 2 \rightsquigarrow \langle\pi_i\rangle\psi_{\pi_i} \text{and } (1 \wedge 2) > \langle\pi\rangle \approx \langle\pi_i\rangle\}$

$\mathit{Spec}_\pi^{12} = \{\phi : \langle\phi, t_\phi\rangle \in \pi^{12} \text{ and } \forall\langle\phi', t_{\phi'}\rangle \in \pi^{12}, t_{\phi'} \not\subset t_\phi \text{ and } \phi \sqcap \psi \neq \bot\}$

$\mathit{Comp}_\pi^{12} = \{\phi : \langle\phi, t_\phi\rangle \in \pi^{12} \text{ and } \forall v \in \mathit{Spec}_\pi^{12}, v \sqcap \phi \neq \bot \text{ and } \phi \sqcap \psi \neq \bot\}$

Spec stands for Specificity; and *Comp* for Completeness.

CON and PDU are used to work out default values on paths, in the case where the reentrancies didn't yield a cycle. CON is analogous to TU1; it ensures that the result of the unification operation in (15) is (16) rather than (17):

(15) $\begin{bmatrix} \mathbf{t}_1^1 \\ \text{F} : /\mathbf{a} \end{bmatrix} \overset{<>}{\sqcap} \begin{bmatrix} \mathbf{t}_2^2 \\ \text{G} : /\mathbf{b} \end{bmatrix}$

(16) $\begin{bmatrix} \mathbf{t}_1 \sqcap \mathbf{t}_2^{(1\wedge 2)} \\ \text{F} : /\mathbf{a} \\ \text{G} : /\mathbf{b} \end{bmatrix}$

(17) $\begin{bmatrix} \mathbf{t}_1 \sqcap \mathbf{t}_2^{(1\wedge 2)} \\ \text{F} : \top \\ \text{G} : \top \end{bmatrix}$

But why is the PDU axiom defined in the way it is? Let's first look at the sets Spec_π^{12} and Comp_π^{12} (we will refer to these as *Spec* and *Comp* when there is no confusion). Spec_π^{12} picks all the values in the tail π^{12} that originated from the most specific TDFSs, and that are compatible with the indefeasible information. This is equivalent to the set (b) in the informal description of $\overset{<>}{\sqcap}$ given earlier. Note for now that *Spec* will help get the specificity results we require. Suppose a tail contains $\langle a, t_a \rangle$ and $\langle b, t_b \rangle$. This means that the TDFS with this tail resulted from the default unification of (at least) the two TDFSs which contained respectively the strictly default information $\pi : a$ and root type t_a, and strictly default information $\pi : b$ and root type t_b. Now suppose that $a \sqcap b = \perp$ and $t_a \sqsubset t_b$. Then $b \notin \mathit{Spec}$. But a will be in *Spec*, as long as (i) t_a was at least as specific as all the other specificity information in the π^{12}, and (ii) a is compatible with the indefeasible information. And if $a \in \mathit{Spec}$, then $b \notin \mathit{Comp}$, since $a \sqcap b = \perp$. So the PDU axiom will ensure that b will be overridden by a, as specificity requires. The Nixon Diamond, therefore, will correspond to a case where *Spec* contains two elements that are incompatible. They must have been associated with types that were unordered with respect to each other. The result of $\overset{<>}{\sqcap}$ in this case will be $\langle \pi \rangle \perp$.

The set Comp^{12}_{π} contains all the values in the tail that are compatible with those in Spec^{12}_{π}—i.e., the values associated with the most specific types—and the indefeasible information. This is equivalent to the set (c) in the informal description of $\overset{<>}{\sqcap}$ given earlier. This enables a value that has been overridden previously by specificity, to now be re-inherited into the result. Therefore, a specific value on a general type constraint can survive unification when it is compatible with the most specific type constraints. As mentioned earlier, allowing this to happen is essential if $\overset{<>}{\sqcap}$ is to reduce to typed unification when no conflicts arise. It is also essential for preserving order independence in the example in §3.2 that illustrated that recording only specificity information from the unification history is insufficient.

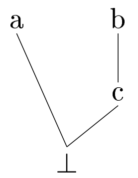
In the definition of *Spec* and *Comp*, we remove all the values that are incompatible with ψ, in order to ensure that indefeasible information always overrides defeasible information, when there is conflict. So, for example, if $a \sqcap b = \perp$, then:

$$\begin{bmatrix} \mathbf{t}_1^1 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{a/c} \end{bmatrix} \end{bmatrix} \overset{<>}{\sqcap} \begin{bmatrix} \mathbf{t}_2^2 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{/b} \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{t}_2^{(1\wedge 2)} \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{a/c} \end{bmatrix} \end{bmatrix} \quad /\{\langle F \cdot G, \{\langle b, t_2\rangle, \langle c, t_1\rangle\}\rangle\}$$

Suppose we didn't remove the information from *Spec* and *Comp* that is incompatible with the indefeasible information, and suppose $t_2 \not\sqsubset t_1$ and $t_1 \not\sqsubset t_2$. Then *Spec* would be $\{b, c\}$, and so the PDU axiom would give $(1 \wedge 2) \Rightarrow \langle F\rangle\langle G\rangle\perp$, rather than $(1 \wedge 2) \Rightarrow \langle F\rangle\langle G\rangle c$. Conflicts between values of equal specificity would thus result in failure, despite the indefeasible evidence which can resolve them.

Finally, the indefeasible information unifies with *Spec* and *Comp* ensuring that the result produces a well-defined TDFS. This information is the unification of all the indefeasible values on paths that, by default, are reentrant with $\langle \pi \rangle$. This looks mysterious at first. The motivation for it is illuminated by the following considerations. First, we must unify *Spec* and *Comp* with (at least) the indefeasible information ψ_π such that $(1 \wedge 2) \rightsquigarrow \langle \pi \rangle \psi_\pi$. To see this consider the following:

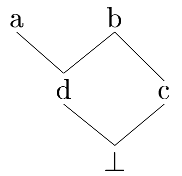


$$\begin{bmatrix} \mathbf{t}^1 & \\ \mathrm{F} : & \mathbf{a} \\ \mathrm{G} : & \mathbf{b} \end{bmatrix} /\{\} \overset{<>}{\sqcap} \begin{bmatrix} \mathbf{t}^2 & \\ \mathrm{F} : & /\boxed{1}\,\mathbf{c} \\ \mathrm{G} : & /\boxed{1} \end{bmatrix} /\{\langle F, \{\langle c, t\rangle\}\rangle, \langle G, \{\langle c, t\rangle\}\rangle\}$$

If the PDU axiom were defined so that $(1 \wedge 2) \Rightarrow \langle F\rangle(Spec_F^{12} \sqcap Comp_F^{12})$, then since in the above $F^{12} = \{\langle c, t\rangle\}$ and $(1 \wedge 2) \rightsquigarrow \langle F\rangle a$, $Spec^{12} = \emptyset$ and $Comp^{12} = \emptyset$. So $(1 \wedge 2) \Rightarrow \langle F\rangle\top$. But by PI2, $\top$ is the most specific value on the defeasible F-path, and by REV, we must infer $(1 \wedge 2) \Rightarrow \langle G\rangle\top$. So for the result to translate back into a well-defined TDFS, we must have $\top \sqsubseteq a$, and $\top \sqsubseteq b$, since the values on defeasible paths must entail those on indefeasible paths. Therefore, $\top \sqsubseteq a \sqcap b$. So $\top \sqsubseteq \bot$. This contradicts our type hierarchy. So, we must unify the indefeasible information ψ_π with *Spec* and *Comp* in the PDU axiom, to ensure the result can be translated back into a well-defined TDFS.

In fact, we unify much more with *Spec* and *Comp*. The reason we need to unify *Spec* and *Comp* with ψ as defined above is because the value on the default path must be more specific than *every* indefeasible value on the paths with which it is reentrant. Otherwise, the result again will not be a well-defined TDFS. Unifying *Spec* and *Comp* with just ψ_π, such that $(1 \wedge 2) \rightsquigarrow \langle\pi\rangle\psi_\pi$ does not guarantee this. If PDU were defined so that $(1 \wedge 2) \Rightarrow \langle F\rangle(Spec_F^{12} \sqcap Comp_F^{12} \sqcap \psi_F)$, where $(1 \wedge 2) \rightsquigarrow \langle F\rangle\psi_F$, then in the above example, we would infer $(1 \wedge 2) \Rightarrow \langle F\rangle\bot$, and $(1 \wedge 2) \Rightarrow \langle G\rangle c$. So by REV and PI2, $c = \bot$, contradicting our hypothesis about the type hierarchy.

To see that we must remove all values from *Spec* and *Comp* that are inconsistent with ψ, consider the above example of $\overset{<>}{\sqcap}$, but with the following type hierarchy:



If we only removed values from $Spec_\pi^{12}$ and $Comp_\pi^{12}$ that were incompatible with ψ_π, then PDU would yield $(1 \wedge 2) \Rightarrow \langle F\rangle(a \sqcap b)$, and $(1 \wedge 2) \Rightarrow \langle G\rangle(a \sqcap b \sqcap c)$. So, via REV and the prime implicate axiom PI2, $d = \bot$, contradicting our hypothesis.

The Tag Axioms

Finally, we need to provide axioms on the tags or indices, that ensure that when the logic predicts that 1 and 2 have identical paths and reentrancies, they are the same tag, and hence name the same TDFS. This is done as follows:

TAG `The Tag Axiom`

- If for all π, π' and for all ϕ:
  $(1 \rightsquigarrow \langle\pi\rangle\phi) \leftrightarrow (2 \rightsquigarrow \langle\pi\rangle\phi)$
  $(1 \rightarrow \langle\pi\rangle \approx \langle\pi'\rangle) \leftrightarrow (2 \rightsquigarrow \langle\pi\rangle \approx \langle\pi'\rangle)$
  $(1 \Rightarrow \langle\pi\rangle\phi) \leftrightarrow (2 \Rightarrow \langle\pi\rangle\phi)$
  $(1 > \langle\pi\rangle \approx \langle\pi'\rangle) \leftrightarrow (2 > \langle\pi\rangle \approx \langle\pi'\rangle)$
  Then
  $\hat{1} = \hat{2}$

We require this, in order to ensure that $\overset{<>}{\sqcap}$ is a function, and is commutative, as we will demonstrate below. The tag axioms ensure that if the logic produces exactly the same paths, then the TDFS they correspond to is the same.

5.4 Examples

In §5.5 we will prove that the above axioms make $\overset{<>}{\sqcap}$ order independent, and also ensure that $\overset{<>}{\sqcap}$ reduces to $\sqcap$ when there's no conflict between the TDFSs. But first, we will illuminate how the above logic works, by means of specific examples. In each case, the task is to translate the TDFSs to be unified into the conditional logic, and then to use the above axioms to generate new $\rightsquigarrow$, $\rightarrow$, $>$ and $\Rightarrow$ rules, which will be translated back into TDFS notation via the inverse translation function.

To demonstrate the equivalence between the informal description of $\overset{<>}{\sqcap}$ given earlier and the formal specification given here, we use the same examples as earlier.

The Defeat of Defeasible Modus Ponens

Consider the following example of $\overset{<>}{\sqcap}$: Where $t_2 \sqsubset t_1$ and $a \sqcap b = \bot$

$$\begin{bmatrix} \mathbf{t}_1^1 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{a} \end{bmatrix} \end{bmatrix} /\{\} \quad \overset{<>}{\sqcap} \quad \begin{bmatrix} \mathbf{t}_2^2 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : /\mathbf{b} \end{bmatrix} \end{bmatrix} /\{\langle F \cdot G, \{\langle b, t_2 \rangle\}\rangle\}$$

The above TDFSs are translated into the following formulae of $\mathcal{L}_{pdu}$:

$$\begin{array}{ll} 1 \rightsquigarrow & t_1 \\ 1 \rightsquigarrow & \langle F \rangle \langle G \rangle a \\ 1 \Rightarrow & t_1 \\ 1 \Rightarrow & \langle F \rangle \langle G \rangle a \\ 2 \rightsquigarrow & t_2 \\ 2 \rightsquigarrow & \langle F \rangle \langle G \rangle \top \\ 2 \Rightarrow & t_2 \\ 2 \Rightarrow & \langle F \rangle \langle G \rangle b \end{array}$$

Using the axioms, we gain the following results:
$(1 \wedge 2) \rightsquigarrow \quad \langle F \rangle \langle G \rangle a$ TU1 and PI1′
By the definition of tails, $Spec^{12}_{F \cdot G}$ and $Comp^{12}_{F \cdot G}$ we have:

$$\begin{array}{rl} F \cdot G^{12} = & \{\langle b, t_2 \rangle\} \\ Spec^{12}_{F \cdot G} = Comp^{12}_{F \cdot G} = & \emptyset \end{array}$$

So
$(1 \wedge 2) \Rightarrow \quad \langle F \rangle \langle G \rangle a$ PDU
Therefore:

$$\begin{aligned} \begin{bmatrix} \mathbf{t}_1^1 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{a} \end{bmatrix} \end{bmatrix} /\{\} \quad \overset{<>}{\sqcap} \quad & \begin{bmatrix} \mathbf{t}_2^2 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : /\mathbf{b} \end{bmatrix} \end{bmatrix} /\{\langle F \cdot G, \{\langle b, t_2 \rangle\}\rangle\} \\ = & \begin{bmatrix} \mathbf{t}_2^{(1 \wedge 2)} \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{a} \end{bmatrix} \end{bmatrix} /\{\langle F \cdot G, \{\langle b, t_2 \rangle\}\rangle\} \end{aligned}$$

Specificity

Consider the following example of $\overset{<>}{\sqcap}$, where $t_2 \sqsubset t_1$, and $a \sqcap b = \bot$.

$$\begin{bmatrix} \mathbf{t}_1^1 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : /\mathbf{a} \end{bmatrix} \end{bmatrix} /\{\langle F \cdot G, \{\langle a, t_1 \rangle\}\rangle\} \overset{<>}{\sqcap} \begin{bmatrix} \mathbf{t}_2^2 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : /\mathbf{b} \end{bmatrix} \end{bmatrix} /\{\langle F \cdot G, \{\langle b, t_2 \rangle\}\rangle\}$$

The above TDFSs are translated into the following formulae of $\mathcal{L}_{pdu}$:

$$\begin{array}{ll} 1 \rightsquigarrow & t_1 \\ 1 \rightsquigarrow & \langle F \rangle \langle G \rangle \top \\ 1 \Rightarrow & t_1 \\ 1 \Rightarrow & \langle F \rangle \langle G \rangle a \\ 2 \rightsquigarrow & t_2 \\ 2 \rightsquigarrow & \langle F \rangle \langle G \rangle \top \\ 2 \Rightarrow & t_2 \\ 2 \Rightarrow & \langle F \rangle \langle G \rangle b \end{array}$$

And the tails are defined as follows:

$$\begin{array}{ll} F \cdot G^1 = & \{\langle a, t_1 \rangle\} \\ F \cdot G^2 = & \{\langle b, t_2 \rangle\} \text{ so} \\ F \cdot G^{12} = & \{\langle a, t_1 \rangle, \langle b, t_2 \rangle\} \\ Spec^{12}_{F \cdot G} = & \{b\} \\ Comp^{12}_{F \cdot G} = & \{b\} \end{array}$$

Using the axioms, we gain the following results:

$$\begin{array}{lll} (1 \wedge 2) \rightsquigarrow & t_2 & \text{TU2 and PI1}' \\ (1 \wedge 2) \rightsquigarrow & \langle F \rangle \langle G \rangle \top & \text{TU2 and PI1}' \\ (1 \wedge 2) \Rightarrow & t_2 & \text{PDU} \\ (1 \wedge 2) \Rightarrow & \langle F \rangle \langle G \rangle b & \text{PDU} \end{array}$$

Therefore:

$$\begin{aligned} &\begin{bmatrix} \mathbf{t}_1^1 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : /\mathbf{a} \end{bmatrix} \end{bmatrix} /\{\langle F \cdot G, \{\langle a, t_1 \rangle\}\rangle\} \overset{<>}{\sqcap} \begin{bmatrix} \mathbf{t}_2^2 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : /\mathbf{b} \end{bmatrix} \end{bmatrix} /\{\langle F \cdot G, \{\langle b, t_2 \rangle\}\rangle\} \\ &= \begin{bmatrix} \mathbf{t}_2^{(1 \wedge 2)} \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : /\mathbf{b} \end{bmatrix} \end{bmatrix} /\{\langle F \cdot G, \{\langle a, t_1 \rangle, \langle b, t_2 \rangle\}\rangle\} \end{aligned}$$

General Values on Specific TDFSs

In §3.2 we gave an example to motivate the need to record the original specificity information for each feature : value path during the unification process. Here we go through an equivalent case in detail:

(1) $\begin{bmatrix} \mathbf{t}_1^1 \\ \mathrm{F} : /\mathbf{a} \end{bmatrix} /\{\langle F, \{\langle a, t_1 \rangle\}\rangle\}$ (2) $\begin{bmatrix} \mathbf{t}_2^2 \\ \mathrm{F} : /\mathbf{b} \end{bmatrix} /\{\langle F, \{\langle b, t_2 \rangle\}\rangle\}$

(3) $\begin{bmatrix} \mathbf{t}_3^3 \\ \mathrm{F} : \mathbf{c} \end{bmatrix} /\{\}$

$$t_3 \sqsubset t_2 \sqsubset t_1$$
$$a \sqcup b = c$$
$$a \sqcap b = \bot$$

We now show that under the above definition of $\overset{<>}{\sqcap}$, the result is order independent. We will calculate $(1 \overset{<>}{\sqcap} 2) \overset{<>}{\sqcap} 3$, and $(1 \overset{<>}{\sqcap} 3) \overset{<>}{\sqcap} 2$.

First, the translations of the above TDFSs produce the following:

$$\begin{array}{ll} 1 \Rightarrow & \langle F \rangle a \\ 2 \Rightarrow & \langle F \rangle b \\ 3 \rightsquigarrow & \langle F \rangle c \end{array}$$

And the tails are defined as follows:

$$\begin{array}{ll} F^1 = & \{\langle a, t_1 \rangle\} \\ F^2 = & \{\langle b, t_2 \rangle\} \\ F^3 = & \emptyset \end{array}$$

We first calculate $1 \overset{<>}{\sqcap} 2$. $F^{12} = \{\langle a, t_1 \rangle, \langle b, t_2 \rangle\}$. And by TU2 and PI1$'$, $(1 \wedge 2) \rightsquigarrow \langle F \rangle \top$; that is the path $F : \top$ is indefeasible information. So $Spec^{12}_F = \{b\}$, and $Comp^{12}_F = \{b\}$. And therefore by the PDU axiom, $(1 \wedge 2) \Rightarrow \langle F \rangle b$.

Now we calculate $(1 \overset{<>}{\sqcap} 2) \overset{<>}{\sqcap} 3$. $F^{(12)3} = \{\langle a, t_1 \rangle, \langle b, t_2 \rangle\}$, and by TU2 and PI1$'$ $((1 \wedge 2) \wedge 3) \rightsquigarrow \langle F \rangle c$. So $Spec^{(12)3}_F = \{b\}$ and $Comp^{(12)3}_F = \{b\}$, and by the PDU axiom $((1 \wedge 2) \wedge 3) \Rightarrow \langle F \rangle b$. So:

$$(1 \overset{<>}{\sqcap} 2) \overset{<>}{\sqcap} 3 = \begin{bmatrix} \mathbf{t}_3^{(12)3} \\ \mathrm{F} : \mathbf{c}/\mathbf{b} \end{bmatrix} \;/\{\langle F, \{\langle a, t_1 \rangle, \langle b, t_2 \rangle\}\rangle\}$$

Now we calculate $(1 \overset{<>}{\sqcap} 3) \overset{<>}{\sqcap} 2$. By the definition of tails, $F^{13} = \{\langle a, t_1 \rangle\}$. And by TU2 and PI1$'$, $(1 \wedge 3) \rightsquigarrow \langle F \rangle c$. So $Spec^{13}_F = \{a\}$, and $Comp^{13}_F = \{a\}$. And so by PDU, $(1 \overset{<>}{\sqcap} 3) \Rightarrow \langle F \rangle a$.

Now, $F^{(13)2} = \{\langle a, t_1 \rangle, \langle b, t_2 \rangle\}$. And by TU2 and PI1$'$ $((1 \wedge 3) \wedge 2) \rightsquigarrow \langle F \rangle c$. So $Spec^{(13)2}_F = \{b\}$ (since $t_2 \sqsubset t_1$), and $Comp^{(13)2}_F = \{b\}$. So by the PDU axiom, $((1 \wedge 3) \wedge 2) \Rightarrow \langle F \rangle b$. So again:

$$(1 \overset{<>}{\sqcap} 3) \overset{<>}{\sqcap} 2 = \begin{bmatrix} \mathbf{t}_3^{(12)3} \\ \mathrm{F} : \mathbf{c}/\mathbf{b} \end{bmatrix} \;/\{\langle F, \{\langle a, t_1 \rangle, \langle b, t_2 \rangle\}\rangle\}$$

Recording Specificity and Values

We now turn to the example introduced in §3.2, which demonstrated that keeping track of only the specificity information of the original default paths was insufficient to guarantee order independence:

$$(1) \begin{bmatrix} \mathbf{t}_1^1 \\ \mathrm{F} : /\mathbf{a} \end{bmatrix} \;/\{\langle F, \{\langle a, t_1 \rangle\}\rangle\} \qquad (2) \begin{bmatrix} \mathbf{t}_2^2 \\ \mathrm{F} : /\mathbf{b} \end{bmatrix} \;/\{\langle F, \{\langle b, t_2 \rangle\}\rangle\}$$

$$(3) \begin{bmatrix} \mathbf{t}_3^3 \\ \mathrm{F} : /\mathbf{d} \end{bmatrix} \;/\{\langle F, \{\langle d, t_3 \rangle\}\rangle\}$$

$$t_3 \sqsubset t_1 \sqsubset t_2$$

$$\begin{array}{c} a \sqsubset b \\ d \sqcap b = c \\ d \sqcap a = \bot \end{array}$$

We show that our logic for $\overset{<>}{\sqcap}$ guarantees order independence for this example. As before, we calculate $(1 \overset{<>}{\sqcap} 2) \overset{<>}{\sqcap} 3$ and $(2 \overset{<>}{\sqcap} 3) \overset{<>}{\sqcap} 1$.

First, the translations of the above TDFSs produce the following:

$$\begin{array}{ll} 1 \Rightarrow & \langle F \rangle a \\ 2 \Rightarrow & \langle F \rangle b \\ 3 \Rightarrow & \langle F \rangle d \end{array}$$

By the definition of tails, $F^{12} = \{\langle a, t_1 \rangle, \langle b, t_2 \rangle\}$, and by TU2 and PI1$'$, $(1 \wedge 2) \rightsquigarrow \langle F \rangle \top$. So $\mathit{Spec}_F^{12} = \{a\}$, and therefore $\mathit{Comp}_F^{12} = \{a, b\}$. So by the PDU axiom, $(1 \wedge 2) \Rightarrow \langle F \rangle a$.

Now, $F^{(12)3} = \{\langle a, t_1 \rangle, \langle b, t_2 \rangle, \langle d, t_3 \rangle\}$, and by TU2 and PI1$'$, $((1 \wedge 2) \wedge 3) \rightsquigarrow \langle F \rangle \top$. So $\mathit{Spec}_F^{(12)3} = \{d\}$ and therefore, $\mathit{Comp}_F^{(12)3} = \{d, b\}$. So by the PDU axiom, $((1 \wedge 2) \wedge 3) \Rightarrow \langle F \rangle c$.

Now we calculate $(2 \overset{<>}{\sqcap} 3) \overset{<>}{\sqcap} 1$. By the definition of tails, $F^{23} = \{\langle b, t_2 \rangle, \langle d, t_3 \rangle\}$. And by TU2 and PI1$'$, $(2 \wedge 3) \rightsquigarrow \langle F \rangle \top$. So $\mathit{Spec}_F^{23} = \{d\}$ and therefore, $\mathit{Comp}_F^{23} = \{b, d\}$. So by the PDU axiom, $(2 \wedge 3) \Rightarrow \langle F \rangle c$.

Again by the definition of tails, $F^{(23)1} = \{\langle b, t_2 \rangle, \langle d, t_3 \rangle, \langle a, t_1 \rangle\}$. And by TU2 and PI1$'$, $((2 \wedge 3) \wedge 1) \rightsquigarrow \langle F \rangle \top$. So $\mathit{Spec}_F^{(23)1} = \{d\}$, and therefore, $\mathit{Comp}_F^{(23)1} = \{b, d\}$. So by the PDU axiom, $((2 \wedge 3) \wedge 1) \Rightarrow \langle F \rangle c$. Hence:

$$(1 \overset{<>}{\sqcap} 2) \overset{<>}{\sqcap} 3 = (2 \overset{<>}{\sqcap} 3) \overset{<>}{\sqcap} 1 = \begin{bmatrix} \mathbf{t}_3^{(13)2} \\ \mathrm{F} : /\mathbf{c} \end{bmatrix} /\{\langle F, \{\langle a, t_1 \rangle, \langle b, t_2 \rangle, \langle d, t_3 \rangle\}\rangle\}$$

Reentrancy

We now examine a simple example involving reentrancy:

$$a \sqcap b = \bot$$

$$\begin{bmatrix} \mathbf{t}^1 \\ \mathrm{F} : \mathbf{a} \\ \mathrm{G} : \mathbf{b} \end{bmatrix} /\{\} \overset{<>}{\sqcap} \begin{bmatrix} \mathbf{t}^2 \\ \mathrm{F} : /\boxed{1} \\ \mathrm{G} : /\boxed{1} \end{bmatrix} /\{\} = \begin{bmatrix} \mathbf{t}^{1 \wedge 2} \\ \mathrm{F} : \mathbf{a}/\boxed{1}\bot \\ \mathrm{G} : \mathbf{b}/\boxed{1} \end{bmatrix} /\{\}$$

When translating the input TDFSs, we obtain the following formulae:

$$\begin{array}{ll} 1 \rightsquigarrow & \langle F \rangle a \\ 1 \rightsquigarrow & \langle G \rangle b \\ 2 > & \langle F \rangle \approx \langle G \rangle \end{array}$$

And the tails are: $F^1 = G^1 = F^2 = G^2 = \emptyset$.

The inferences are as follows:

$$\begin{array}{ll} (1 \wedge 2) > \langle F \rangle \approx \langle G \rangle & \text{RE2} \\ (1 \wedge 2) \rightsquigarrow \langle F \rangle a & \text{TU2 and PI1}' \\ (1 \wedge 2) \rightsquigarrow \langle G \rangle b & \text{TU2 and PI1}' \end{array}$$

Furthermore, calculating the δ such that $(1 \wedge 2) \Rightarrow \langle F \rangle \delta$ involves applying the PDU axiom. that is, δ is calculated via the tails. $F^{12} = G^{12} = \emptyset$. So $\mathit{Spec}_F^{12} = \mathit{Comp}_F^{12} = \emptyset$. So by the PDU axiom, $\delta = \mathit{Spec} \sqcap \mathit{Comp} \sqcap \psi$, where $\psi = \sqcap\{a, b\}$. Hence by PDU, $(1 \wedge 2) \Rightarrow \langle F \rangle \bot$. Hence $1 \overset{<>}{\sqcap} 2$ is as defined above.

5.5 Theorems and Proofs

We have so far given the axioms of the logic in which $\overset{<>}{\sqcap}$ is axiomatised, and we have illustrated how the logic works via some simple examples. We should now prove that the version of $\overset{<>}{\sqcap}$ that the logic defines does indeed have the properties we desire.

First, we define precisely the logical consequence relation $\mid\!\sim_{pdu}$ in terms of which $\overset{<>}{\sqcap}$ is defined:

Definition 7 PDU Deduction $\mid\!\sim_{pdu}$

A PDU deduction T from the translation of two typed default feature structures FS_1 *and* FS_2*—written respectively as* $\tau(\mathrm{FS}_1)$ *and* $\tau(\mathrm{FS}_2)$*—is a sequence of lines in which every line of* $\tau(\mathrm{FS}_1) \cup \tau(\mathrm{FS}_2)$ *occurs and for every formula of the form* $1 \Rightarrow \langle\pi\rangle\phi$ *or* $2 \Rightarrow \langle\pi\rangle\phi$*, there is a formula of the form* $(1 \wedge 2) \Rightarrow \langle\pi\rangle\chi'$ *as a line in T, and similarly for* $\rightarrow$, $\rightsquigarrow$ *and* $>$*; or* $(1 \wedge 2) > \bot$ *and* $(1 \wedge 2) \not\rightarrow \bot$ *and the above conditions hold for just* $\rightarrow$ *and* $\rightsquigarrow$*; or* $(1 \wedge 2) \rightarrow \bot$ *and* $(1 \wedge 2) > \bot$*.*

Definition 8 $FS_1 \overset{<>}{\sqcap} FS_2 = FS_3$ *iff*
$\tau(FS_1), \tau(FS_2) \mid\!\sim_{pdu} \tau(FS_3)$*.*

Note that $\mid\!\sim_{pdu}$ is neither closed on the right, nor supraclassical:

- $\Gamma_1, \Gamma_2 \mid\!\sim_{pdu} \Gamma_3$ and $\Gamma_3 \models \Gamma_4$ doesn't entail that $\Gamma_1, \Gamma_2 \mid\!\sim_{pdu} \Gamma_4$.
- $\Gamma_1, \Gamma_2 \models \Gamma_3$ doesn't entail that $\Gamma_1, \Gamma_2 \mid\!\sim_{pdu} \Gamma_3$.

Also note that if for one reason or another one can infer $(1 \wedge 2) \rightarrow \bot$ or $(1 \wedge 2) > \bot$ (e.g., the reentrancies in 1 and 2 yield a cycle in the indefeasible part or defeasible part respectively), then the corresponding TFS in the result will be $\bot$.

Using the above definitions, we prove the following lemmas and theorems:

Lemma 1 *If* $1 \Rightarrow \langle\pi\rangle a$ *then* $a = \ Spec^1_\pi \sqcap Comp^1_\pi \sqcap \psi$ *where*
$\psi = \sqcap\{\psi' : 1 \rightsquigarrow \langle\pi'\rangle\psi' \text{ and } 1 > \langle\pi\rangle \approx \langle\pi'\rangle\}$

Note that this lemma implies that if $(1 \wedge 2) \Rightarrow \langle\pi\rangle\delta$, then δ is determined by the tails, and not the defeasible TFS of the TDFSs. We now prove this lemma.

Proof

By induction on the number of unifications to get 1.

Base case: Where 1 is a TDFS in the initial KB, either:

1. $1 \rightsquigarrow \langle\pi\rangle a$ and $\pi^1 = \emptyset$; or
2. $1 \rightsquigarrow \langle\pi\rangle\psi_\pi$, $a \sqsubset \psi_\pi$, and $\pi^1 = \{\langle a, t\rangle\}$, where $1 \Rightarrow t$.

Consider the first case. Then $\psi = a$ because for all π' such that $1 > \langle\pi\rangle \approx \langle\pi'\rangle$ and $1 \rightsquigarrow \langle\pi'\rangle\psi'$, $a \sqsubseteq \psi'$. For suppose not, then there exists a ψ', such that $1 \rightsquigarrow \langle\pi'\rangle\psi'$, $1 \Rightarrow \langle\pi'\rangle a$ by REV, and $a \not\sqsubseteq \psi'$. This contradicts the definition of TDFSs, and the way they are translated into $\mathcal{L}_{pdu}$. So $\psi = a$. So since $\pi^1 = \emptyset$, $a = Spec^1_\pi \sqcap Comp^1_\pi \sqcap \psi$, as required.

Now consider the second case. Then by REV, the definition of TDFSs, and the translation of TDFSs into $\mathcal{L}_{pdu}$, we know that for all ψ' such that $1 \rightsquigarrow \langle\pi'\rangle\psi'$ and $1 > \langle\pi\rangle \approx \langle\pi'\rangle$, $a \sqsubseteq \psi'$.

So $a \sqsubseteq \psi$. And since $\pi^1 = \{\langle a, t \rangle\}$, $Spec^1_\pi = Comp^1_\pi = \{a\}$. So $a = Spec^1_\pi \sqcap Comp^1_\pi \sqcap \psi$, as required.

The Inductive Step: Assume the result holds for n unifications, and 1 is obtained by $n+1$ unifications. So $1 = 3 \wedge 4$, where 3 and 4 where obtained by $\leq n$ unifications each. Now, $1 \Rightarrow \langle\pi\rangle\delta$ must have been derived from 3 and 4 by CON or PDU. If it was derived by PDU, then the result holds by the definition of PDU.

So suppose 1 were derived by CON. Then:

$$\begin{array}{l} 3 \Rightarrow \langle\pi\rangle a \\ (3 > \langle\pi\rangle \approx \langle\pi'\rangle) \rightarrow (4 \not> \langle\pi'\rangle\top) \\ (3 \wedge 4) \not> \langle\pi_1\pi_2\rangle \approx \langle\pi_2\rangle \end{array}$$

So if $3 > \langle\pi\rangle \approx \langle\pi'\rangle$, then $\pi'^4 = \emptyset$, and the reentrancies in 3 and 4 don't produce a cycle. By the inductive hypothesis:

$$\begin{array}{l} a = Spec^3_\pi \sqcap Comp^3_\pi \sqcap \psi^3, \\ \text{where} \\ \psi^3 = \sqcap\{\psi'^3 : 3 \rightsquigarrow \langle\pi'\rangle\psi'^3 \text{ and } 3 > \langle\pi\rangle \approx \langle\pi'\rangle\} \end{array}$$

By definition:

$$\pi^{34} = \cup^{\pi'}_{(3\wedge4)>\langle\pi\rangle\approx\langle\pi'\rangle}\pi'^3 \cup \pi'^4$$

We now show that $(3 \wedge 4) > \langle\pi\rangle \approx \langle\pi'\rangle$ iff $3 > \langle\pi\rangle \approx \langle\pi'\rangle$. The if-condition follows from RE2. Now consider the only if condition. Suppose that $(3 \wedge 4) > \langle\pi\rangle \approx \langle\pi'\rangle$ and $3 \not> \langle\pi\rangle \approx \langle\pi'\rangle$. By the following premise of CON—$(3 \wedge 4) \not> \langle\pi_1\pi_2\rangle \approx \langle\pi_2\rangle$—we can assume that $3 \wedge 4$ doesn't yield cyclic reentrancies. So, there must be a π_1 and π_2, where $\pi_1 \neq \pi_2$, and $3 > \langle\pi\rangle \approx \langle\pi_1\rangle$, $4 > \langle\pi_1\rangle \approx \langle\pi_2\rangle$, and $3 > \langle\pi_2\rangle \approx \langle\pi'\rangle$. For otherwise by RE2, $(3 \wedge 4) \not> \langle\pi\rangle \approx \langle\pi'\rangle$. But this contradicts our assumptions, because $3 > \langle\pi\rangle \approx \langle\pi_1\rangle$, and $4 \Rightarrow \langle\pi_1\rangle\phi$, where $\phi \sqsubseteq \top$. So $(3 \wedge 4) > \langle\pi\rangle \approx \langle\pi'\rangle$ only if $3 > \langle\pi\rangle \approx \langle\pi'\rangle$. So

$$\{\pi' : (3 \wedge 4) > \langle\pi\rangle \approx \langle\pi'\rangle\} = \{\pi' : 3 > \langle\pi\rangle \approx \langle\pi'\rangle\}$$

And by assumption, for all π' such that $3 > \langle\pi\rangle \approx \langle\pi'\rangle$, $\pi'^4 = \emptyset$. So

$$\pi^{34} = \cup^{\pi'}_{3>\langle\pi\rangle\approx\langle\pi'\rangle}\pi'^3 = \pi^3$$

Now consider π' such that $3 > \langle\pi\rangle \approx \langle\pi'\rangle$. Then $(3 \wedge 4) \rightsquigarrow \langle\pi'\rangle\psi'$ iff $3 \rightsquigarrow \langle\pi'\rangle\psi'$, because by the premises, $4 \not\rightsquigarrow \langle\pi'\rangle\top$. So:

$$\begin{array}{l} \{\psi' : (3 \wedge 4) \rightsquigarrow \langle\pi'\rangle\psi' \text{ and } (3 \wedge 4) > \langle\pi\rangle \approx \langle\pi'\rangle\} = \{\psi' : 3 \rightsquigarrow \langle\pi'\rangle\psi' \text{ and } 3 > \langle\pi\rangle \approx \langle\pi'\rangle\} \\ Spec^{34}_\pi = Spec^3_\pi \\ Comp^{34}_\pi = Comp^3_\pi \end{array}$$

So

$$a = Spec^1_\pi \sqcap Comp^1_\pi \sqcap \psi$$

as required. □

Lemma 2 $\overset{<>}{\sqcap}$ *is a function. That is,* $\Gamma_1 \overset{<>}{\sqcap} \Gamma_2$ *is a unique* TDFS.

Proof

Suppose not. Then $\tau(\Gamma_1)$ and $\tau(\Gamma_2)$ yield either

1. $(1 \wedge 2) > \langle\pi\rangle \approx \langle\pi'\rangle$ and $(1 \wedge 2) \not> \langle\pi\rangle \approx \langle\pi'\rangle$; or
2. $(1 \wedge 2) \Rightarrow \langle\pi\rangle\phi$, $(1 \wedge 2) \Rightarrow \langle\pi\rangle\psi$, and $\phi \neq \psi$; or
3. $(1 \wedge 2) \rightsquigarrow \langle\pi\rangle a$, $(1 \wedge 2) \Rightarrow \langle\pi\rangle b$, and $b \not\sqsubseteq a$; or
4. $(1 \wedge 2) \rightarrow \langle\pi\rangle \approx \langle\pi'\rangle$ and $(1 \wedge 2) \not> \langle\pi\rangle \approx \langle\pi'\rangle$.

Case 1 is impossible by the axioms of the logic. Case 2 isn't possible by lemma 1. Case 3 is impossible by lemma 1, and case 4 is impossible by RE1. □

Lemma 3 $\overset{<>}{\sqcap}$ *is commutative.*

Proof
PDU, CON, TU1, TU2 and the reentrancy axioms do not distinguish the commutative order. So for all π, π', and ϕ:

$$\begin{array}{lll}(1 \wedge 2) \Rightarrow \langle\pi\rangle\phi & \text{iff} & (2 \wedge 1) \Rightarrow \langle\pi\rangle\phi \\ (1 \wedge 2) > \langle\pi\rangle \approx \langle\pi'\rangle & \text{iff} & (2 \wedge 1) > \langle\pi\rangle \approx \langle\pi'\rangle \\ (1 \wedge 2) \rightsquigarrow \langle\pi\rangle\phi & \text{iff} & (2 \wedge 1) \rightsquigarrow \langle\pi\rangle\phi \\ (1 \wedge 2) \rightarrow \langle\pi\rangle \approx \langle\pi'\rangle & \text{iff} & (2 \wedge 1) \rightarrow \langle\pi\rangle \approx \langle\pi'\rangle\end{array}$$

So by the tag axioms, $\widehat{1 \wedge 2} = \widehat{2 \wedge 1}$. And therefore $\Gamma_1 \overset{<>}{\sqcap} \Gamma_2$ and $\Gamma_2 \overset{<>}{\sqcap} \Gamma_1$ have identical paths, reentrancies and tags.

Furthermore,

$$\begin{array}{rcl}\pi^{12} & = & \bigcup^{\pi'}_{(1\wedge2)>\langle\pi\rangle\approx\langle\pi'\rangle} \pi'^1 \cup \pi'^2 \\ & = & \bigcup^{\pi'}_{(2\wedge1)>\langle\pi\rangle\approx\langle\pi'\rangle} \pi'^2 \cup \pi'^1 \\ & = & \pi^{21}\end{array}$$

So $\Gamma_1 \overset{<>}{\sqcap} \Gamma_2$ and $\Gamma_2 \overset{<>}{\sqcap} \Gamma_1$ also have identical tails. Therefore $\Gamma_1 \overset{<>}{\sqcap} \Gamma_2 = \Gamma_2 \overset{<>}{\sqcap} \Gamma_1$. □

Lemma 4 $\overset{<>}{\sqcap}$ *is associative.*

Proof
We need to show:

1. $((1 \wedge 2) \wedge 3) \rightsquigarrow \langle\pi\rangle\phi$ iff $(1 \wedge (2 \wedge 3)) \rightsquigarrow \langle\pi\rangle\phi$.
2. $((1 \wedge 2) \wedge 3) \rightarrow \langle\pi\rangle \approx \langle\pi'\rangle$ iff $(1 \wedge (2 \wedge 3)) \rightarrow \langle\pi\rangle \approx \langle\pi'\rangle$.
3. $((1 \wedge 2) \wedge 3) > \langle\pi\rangle \approx \langle\pi'\rangle$ iff $(1 \wedge (2 \wedge 3)) > \langle\pi\rangle \approx \langle\pi'\rangle$.
4. $\pi^{(12)3} = \pi^{1(23)}$
5. $((1 \wedge 2) \wedge 3) \Rightarrow \langle\pi\rangle\phi$ iff $(1 \wedge (2 \wedge 3)) \Rightarrow \langle\pi\rangle\phi$.

Cases 1 and 2 hold because typed unification is associative, and the axioms on $\rightsquigarrow$ and $\rightarrow$ model typed unification.

We now show Case 3: that the default reentrancies are associative. So, suppose that $((1 \wedge 2) \wedge 3) > \langle\pi\rangle \approx \langle\pi'\rangle$. Then by RE1 and RE2, this holds iff:

(a) $(1 \wedge 2) > \langle\pi\rangle \approx \langle\pi_1\rangle$, $3 > \langle\pi_1\rangle \approx \langle\pi_2\rangle$, and $(1 \wedge 2) > \langle\pi_2\rangle \approx \langle\pi'\rangle$; or

(b) $3 > \langle\pi\rangle \approx \langle\pi_1\rangle$, $(1 \wedge 2) > \langle\pi_1\rangle \approx \langle\pi_2\rangle$, and $3 > \langle\pi_2\rangle \approx \langle\pi'\rangle$.[12]

Suppose Case (a). Then since $(1 \wedge 2) > \langle\pi\rangle \approx \langle\pi_1\rangle$, this holds iff:

(a) i. $1 > \langle\pi\rangle \approx \langle\pi_3\rangle$, $2 > \langle\pi_3\rangle \approx \langle\pi_4\rangle$, and $1 > \langle\pi_4\rangle \approx \langle\pi_1\rangle$; or

ii. $2 > \langle\pi\rangle \approx \langle\pi_3\rangle$, $1 > \langle\pi_3\rangle \approx \langle\pi_4\rangle$, and $2 > \langle\pi_4\rangle \approx \langle\pi_1\rangle$.

And also, since $(1 \wedge 2) > \langle\pi_2\rangle \approx \langle\pi'\rangle$, case (a) holds iff (iii) or (iv) hold:

(a) iii. $1 > \langle\pi_2\rangle \approx \langle\pi_5\rangle$, $2 > \langle\pi_5\rangle \approx \langle\pi_6\rangle$ and $1 > \langle\pi_6\rangle \approx \langle\pi'\rangle$; or

iv. $2 > \langle\pi_2\rangle \approx \langle\pi_5\rangle$, $1 > \langle\pi_5\rangle \approx \langle\pi_6\rangle$ and $2 > \langle\pi_6\rangle \approx \langle\pi'\rangle$.

Suppose (i) and (iii) hold. Then by RE2 and AND:

$$(2 \wedge 3) > (\langle\pi_3\rangle \approx \langle\pi_4\rangle \wedge \langle\pi_1\rangle \approx \langle\pi_2\rangle \wedge \langle\pi_5\rangle \approx \langle\pi_6\rangle)$$

And by RE2 and AND:

$$\begin{array}{rl}(1 \wedge (2 \wedge 3)) > (& \langle\pi\rangle \approx \langle\pi_3\rangle \wedge \langle\pi_3\rangle \approx \langle\pi_4\rangle \wedge \langle\pi_4\rangle \approx \langle\pi_1\rangle \wedge \\ & \langle\pi_1\rangle \approx \langle\pi_2\rangle \wedge \langle\pi_2\rangle \approx \langle\pi_5\rangle \wedge \langle\pi_5\rangle \approx \langle\pi_6\rangle \wedge \langle\pi_6\rangle \approx \langle\pi'\rangle)\end{array}$$

So by ARE and CR, $(1 \wedge (2 \wedge 3)) > \langle\pi\rangle \approx \langle\pi'\rangle$.

The other permutations of (i–iv) are symmetric to this case. And the case for (b) is symmetric to case (a). So $((1 \wedge 2) \wedge 3) > \langle\pi\rangle \approx \langle\pi'\rangle$ entails $(1 \wedge (2 \wedge 3)) > \langle\pi\rangle \approx \langle\pi'\rangle$. The argument that $(1 \wedge (2 \wedge 3)) > \langle\pi\rangle \approx \langle\pi'\rangle$ entails $((1 \wedge 2) \wedge 3) > \langle\pi\rangle \approx \langle\pi'\rangle$ is similar. Hence reentrancies are associative.

Now we prove 4: $\pi^{(12)3} = \pi^{1(23)}$

$$\begin{array}{rl}\pi^{(12)3} = & \cup^{\pi'}_{((1\wedge2)\wedge3)>\langle\pi\rangle\approx\langle\pi'\rangle} \pi'^{12} \cup \pi'^{3} \\ = & \cup^{\pi'}_{((1\wedge2)\wedge3)>\langle\pi\rangle\approx\langle\pi'\rangle} \pi'^{3} \cup (\cup^{\pi''}_{(1\wedge2)>\langle\pi'\rangle\approx\langle\pi''\rangle} \pi''^{1} \cup \pi''^{2})\end{array}$$

Now if $(1 \wedge 2) > \langle\pi'\rangle \approx \langle\pi''\rangle$, then $((1 \wedge 2) \wedge 3) > \langle\pi'\rangle \approx \langle\pi''\rangle$ by RE2. So if in addition $((1 \wedge 2) \wedge 3) > \langle\pi\rangle \approx \langle\pi'\rangle$, then by ARE and AND, $((1 \wedge 2) \wedge 3) > \langle\pi\rangle \approx \langle\pi''\rangle$. So since $(1 \wedge 2) > \langle\pi'\rangle \approx \langle\pi'\rangle$,

$$\cup^{\pi'}_{((1\wedge2)\wedge3)>\langle\pi\rangle\approx\langle\pi'\rangle} \cup^{\pi''}_{(1\wedge2)>\langle\pi'\rangle\approx\langle\pi''\rangle} \pi''^{1} \cup \pi''^{2} = \cup^{\pi'}_{((1\wedge2)\wedge3)>\langle\pi\rangle\approx\langle\pi'\rangle} \pi'^{1} \cup \pi'^{2}$$

Hence

$$\pi^{(12)3} = \cup^{\pi'}_{((1\wedge2)\wedge3)>\langle\pi\rangle\approx\langle\pi'\rangle} \pi'^{1} \cup \pi'^{2} \cup \pi'^{3}$$

By a similar argument:

$$\pi^{1(23)} = \cup^{\pi'}_{(1\wedge(2\wedge3))>\langle\pi\rangle\approx\langle\pi'\rangle} \pi'^{1} \cup \pi'^{2} \cup \pi'^{3}$$

But $\{\pi' : ((1 \wedge 2) \wedge 3) > \langle\pi\rangle \approx \langle\pi'\rangle\} = \{\pi' : (1 \wedge (2 \wedge 3)) > \langle\pi\rangle \approx \langle\pi'\rangle\}$. So $\pi^{(12)3} = \pi^{1(23)}$.

Now we prove 5: $((1 \wedge 2) \wedge 3) \Rightarrow \langle\pi\rangle\phi$ iff $(1 \wedge (2 \wedge 3)) \Rightarrow \langle\pi\rangle\phi$.

Suppose $((1 \wedge 2) \wedge 3) \Rightarrow \langle\pi\rangle\delta_1$ and $(1 \wedge (2 \wedge 3)) \Rightarrow \langle\pi\rangle\delta_2$. Then by lemma 1:

$$\begin{array}{rl}\delta_1 = & \mathit{Spec}^{(12)3}_{\pi} \sqcap \mathit{Comp}^{(12)3}_{\pi} \sqcap \{\psi_i : ((1 \wedge 2) \wedge 3) > \langle\pi\rangle \approx \langle\pi_i\rangle \text{ and } ((1 \wedge 2) \wedge 3) \rightsquigarrow \langle\pi_i\rangle\psi_i\} \\ = & \mathit{Spec}^{1(23)}_{\pi} \sqcap \mathit{Comp}^{1(23)}_{\pi} \sqcap \{\psi_i : (1 \wedge (2 \wedge 3)) > \langle\pi\rangle \approx \langle\pi_i\rangle \text{ and } (1 \wedge (2 \wedge 3)) \rightsquigarrow \langle\pi_i\rangle\psi_i\} \\ = & \delta_2\end{array}$$

So $\overset{<>}{\sqcap}$ is associative. □

[12] We allow for the possibility here that there are identities between π, π_1, π_2, and π'.

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

Proof
Follows immediately from lemmas 2, 3 and 4.

Theorem 2 The Typed Unification Property
Suppose $(1 \wedge 2) > \langle\pi\rangle \approx \langle\pi_i\rangle$, $1 \leq i \leq n$ *and if* $(1 \wedge 2) > \langle\pi\rangle \approx \langle\pi'\rangle$ *then* $\pi' = \pi_i$ *for some* i, $1 \leq i \leq n$.
Let:

$$\begin{array}{rl} \phi = & \{x_i^1 : 1 \Rightarrow \langle\pi_i\rangle x_i^1, 1 \leq i \leq n\} \cup \\ & \{x_i^2 : 2 \Rightarrow \langle\pi_i\rangle x_i^2, 1 \leq i \leq n\} \end{array}$$

and

$$(1 \wedge 2) \Rightarrow \langle\pi\rangle\delta$$

Then if $\delta \neq \bot$ *and* $\sqcap\phi \neq \bot$, *then* $\delta = \sqcap\phi$.

Proof
By lemma 1, $(1 \wedge 2) \Rightarrow \langle\pi\rangle\delta$, where

$$\begin{array}{rl} \delta = & Spec_\pi^{12} \sqcap Comp_\pi^{12} \sqcap \psi \text{ where} \\ \psi = & \sqcap\{\psi_i : (1 \wedge 2) \rightsquigarrow \langle\pi_i\rangle\psi_i, 1 \leq i \leq n\} \end{array}$$

Furthermore for $1 \leq i \leq n$, by lemma 1:

$$\begin{array}{rl} x_i^1 = & Spec_{\pi_i}^1 \sqcap Comp_{\pi_i}^1 \sqcap \chi_i^1 \text{ where} \\ \chi_i^1 = & \sqcap\{\zeta_j^1 : 1 > \langle\pi_i\rangle \approx \langle\pi_j\rangle \text{ and } 1 \rightsquigarrow \langle\pi_j\rangle\zeta_j^1\} \text{ and} \\ x_i^2 = & Spec_{\pi_i}^2 \sqcap Comp_{\pi_i}^2 \sqcap \chi_i^2 \text{ where} \\ \chi_i^2 = & \sqcap\{\zeta_j^2 : 2 > \langle\pi_i\rangle \approx \langle\pi_j\rangle \text{ and } 2 \rightsquigarrow \langle\pi_j\rangle\zeta_j^1\} \end{array}$$

So by TU1, TU2 PI1$'$, ARE and REV:

$$\psi = \sqcap\{\chi_i^1, \chi_i^2 : 1 \leq i \leq n\})$$

Now

$$\pi^{12} = \cup_{1 \leq i \leq n}\pi_i^1 \cup \pi_i^2$$

We now show $Spec_\pi^{12} \subseteq \cup_{1 \leq i \leq n} Spec_{\pi_i}^1 \cup Spec_{\pi_i}^2$. Suppose that $v \in Spec_\pi^{12}$. Then $\langle v, t_v\rangle \in \pi_i^1$ or $\langle v, t_v\rangle \in \pi_i^2$ for some i, such that t_v is a most specific type in π^{12}. Suppose without loss of generality that $\langle v, t_v\rangle \in \pi_i^1$. Then t_v must be a most specific type in π_i^1. And if $v \sqcap \chi_i^1 = \bot$, then $v \sqcap \psi = \bot$, and so $v \notin Spec_\pi^{12}$. So $v \sqcap \chi_i^1 \neq \bot$. Hence by definition, $v \in Spec_{\pi_i}^1$. So $Spec_\pi^{12} \subseteq \cup_{1 \leq i \leq n} Spec_{\pi_i}^1 \cup Spec_{\pi_i}^2$.

Now consider $v \in \cup_{1 \leq i \leq n} Spec_{\pi_i}^1 \cup Spec_{\pi_i}^2$ where $v \notin Spec_\pi^{12}$. Then since $\sqcap\phi \neq \bot$, $v \sqcap Spec_\pi^{12} \neq \bot$, since by the above $Spec_\pi^{12} \subseteq \phi$ and $v \in \phi$. Furthermore, $v \sqcap \{\chi_i^1, \chi_i^2 : 1 \leq i \leq n\} \neq \bot$, because otherwise $\sqcap\phi = \bot$. So $v \sqcap \psi \neq \bot$. So by definition, $v \in Comp_\pi^{12}$. So

$$\cup_{1 \leq i \leq n} Spec_{\pi_i}^1 \cup Spec_{\pi_i}^2 \subseteq Spec_\pi^{12} \cup Comp_\pi^{12}$$

Now consider $w \in \cup_{1 \leq i \leq n} Comp_{\pi_i}^1 \cup Comp_{\pi_i}^2$. Suppose without loss of generality that $w \in Comp_{\pi_i}^1$. Then if $w \sqcap Spec_{\pi_j}^1 = \bot$ or $w \sqcap Spec_{\pi_j}^2 = \bot$ for some j, $\sqcap\phi = \bot$, contradicting our assumption. So $w \sqcap Spec^{12}_\pi \neq \perp$. Similarly, if $w \sqcap \chi^1_j = \perp$ or $w \sqcap \chi^2_j = \perp$ for some j, then $\sqcap\phi = \perp$. So $w \sqcap \psi \neq \perp$. So $w \in Comp^{12}_\pi$. Hence:

$$\cup_{1\leq i\leq n} Comp^1_{\pi_i} \cup Comp^2_{\pi_i} \subseteq Comp^{12}_\pi$$

Now choose $x \in \pi^1_i$. Suppose $x \notin Comp^1_{\pi_i}$. Then $x \sqcap Spec^1_{\pi_i} = \perp$ or $x \sqcap \chi^1_i = \perp$.

If $x \sqcap Spec^1_{\pi_i} = \perp$, then $x \sqcap \{Spec^1_{\pi_i}, Spec^2_{\pi_i} : 1 \leq i \leq n\} = \perp$. But $Spec^{12}_\pi \sqcap Comp^{12}_\pi \sqsubseteq \sqcap\{Spec^1_{\pi_i}, Spec^2_{\pi_i} : 1 \leq i \leq n\}$, since $Spec^{12}_\pi \cup Comp^{12}_\pi \supseteq \{Spec^1_{\pi_i}, Spec^2_{\pi_i} : 1 \leq i \leq n\}$. So $x \sqcap \delta = \perp$. So if $x \in Comp^{12}_\pi$, then $x \sqcap \delta = \delta = \perp$. This contradicts our hypothesis. So $x \notin Comp^{12}_\pi$.

If $x \sqcap \chi^1_i = \perp$, then $x \sqcap \psi = \perp$, and so $x \sqcap \delta = \perp$. Therefore, if $x \in Comp^{12}_\pi$, then $x \sqcap \delta = \delta = \perp$. So $x \notin Comp^{12}_\pi$.

Hence $\cup_{1\leq i\leq n} Comp^1_{\pi_i} \cup Comp^2_{\pi_i} \supseteq Comp^{12}_\pi$. So:

$$\begin{array}{rcll} \cup_{1\leq i\leq n} Comp^1_{\pi_i} \cup Comp^2_{\pi_i} &=& Comp^{12}_\pi & \text{and therefore} \\ \cup_{1\leq i\leq n} Spec^1_{\pi_i} \cup Comp^1_{\pi_i} \cup Spec^2_{\pi_i} \cup Comp^2_{\pi_i} &=& Spec^{12}_\pi \cup Comp^{12}_\pi & \end{array}$$

So, since $\sqcap\{\chi^1_i, \chi^2_i : 1 \leq i \leq n\} = \psi$,

$$\begin{array}{rl} \sqcap\phi = & Spec^{12}_\pi \sqcap Comp^{12}_\pi \sqcap \psi \\ = & \delta \end{array}$$

□

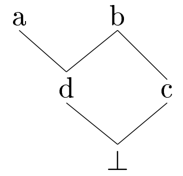
The above theorem confirms that $\overset{<>}{\sqcap}$ reduces to typed unification when no conflict arises. But 'no conflict' doesn't mean 'no conflict in the FS components of the TDFSs'. The crucial extra premise is: $(1 \wedge 2) \Rightarrow \langle\pi\rangle\delta$ where $\delta \neq \perp$. By lemma 1, δ is defined by the tails, and not the FSs.

Without this condition, the theorem would be one where $\overset{<>}{\sqcap}$ reduces to $\sqcap$ when there is no conflict among the FS components of the TDFSs. But this is not a valid result. The following example illustrates this:

$$\begin{bmatrix} \mathbf{t}^1_1 \\ \mathrm{F} : /\mathbf{a} \end{bmatrix} /\{\langle F, \{\langle a, t_1\rangle, \langle c, t_2\rangle\}\rangle\} \overset{<>}{\sqcap} \begin{bmatrix} \mathbf{t}^2_3 \\ \mathrm{F} : /\mathbf{b} \end{bmatrix} /\{\langle F, \{\langle b, t_3\rangle\}\rangle\}$$

Where:

$$t_3 \sqsubset t_1 \sqsubset t_2$$
$$a \sqcap b = d,\ c \sqcap a = \perp,\ c \sqsubset b$$



In the above, the FSs components do not conflict, but the tails do. If the Typed Unification Property held in cases where there was no conflict among the FS components, then we should be able to infer that $(1 \wedge 2) \Rightarrow \langle F\rangle d$ from the axioms.

But this does not hold. By the definition of tails, *Spec* and *Comp*, $Spec^{12}_F = \{b\}$, $Comp^{12}_F = \{a, b, c\}$, and so $Spec^{12}_F \sqcap Comp^{12}_F = \perp$ (and *not* d). Therefore:

$$FS1 \overset{<>}{\sqcap} FS2 = \begin{bmatrix} \mathbf{t}^{12}_3 \\ \mathrm{F} : /\perp \end{bmatrix} /\{\langle F, \{\langle a, t_1\rangle, \langle c, t_2\rangle, \langle b, t_3\rangle\}\rangle\}$$

We could define *Comp* in the PDU axiom, so that *Comp* prioritises the defaults that are compatible with the values in *Spec*. Here, this would mean that a would be in *Comp* but c would not, because c is associated with a more general type than a is, and c is incompatible with a thereby yielding priority to a. The defeasible result on the F path would then be d rather than $\bot$. However, this modification to the PDU axiom, while preserving order independence, increases complexity. It also does not ultimately produce an operation more integrated with typing in general; the reasons for this are discussed in detail in Lascarides and Copestake (1995). It does show, however, that the conditional logic approach to defining order independent $\overset{<>}{\sqcap}$ provides several options.

The Typed Unification Property is not guaranteed when the defeasible FS components do not conflict. But the only added condition here is that $\delta \neq \bot$, which is quite weak. This permits $\overset{<>}{\sqcap}$ to reduce to $\sqcap$ in certain cases where the tails are in conflict.

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 (see §4.3) and we can prove for these cases that the typed unification property is one where $\overset{<>}{\sqcap}$ reduces to $\sqcap$ when there is no conflict among the FS components:

Corollary 1 *If* TDFS*1 and* TDFS2 *are basic* TDFS*s, and* $(1 \wedge 2) > \langle\pi\rangle \approx \langle\pi_i\rangle$ *for* $1 \leq i \leq n$, *and*
if $(1 \wedge 2) > \langle\pi\rangle \approx \langle\pi'\rangle$, *then* $\pi' = \pi_i$ *for some* i,
and $\sqcap\phi \neq \bot$, *where*

$$\begin{aligned} \phi = \; & \{x_i^1 : 1 \Rightarrow \langle\pi_i\rangle x_i^1, 1 \leq i \leq n\} \cup \\ & \{x_i^2 : 2 \Rightarrow \langle\pi_i\rangle x_i^2, 1 \leq i \leq n\} \end{aligned}$$

Then $(1 \wedge 2) \Rightarrow \langle\pi\rangle(\sqcap\phi)$

Proof

By lemma 1, $x_i^1 = Spec^1_{\pi_i} \sqcap Comp^1_{\pi_i} \sqcap \psi_i^1$, where $\psi_i^1 = \{\chi_j^1 : 1 \rightsquigarrow \langle\pi_j\rangle\chi_j^1 \text{ and } 1 \rightsquigarrow \langle\pi_i\rangle \approx \langle\pi_j\rangle\}$. Similarly for 2.

And by lemma 1, $(1 \wedge 2) \Rightarrow \langle\pi\rangle\delta$, where $\delta = Spec^{12}_{\pi} \sqcap Comp^{12}_{\pi} \sqcap \psi$, and $\psi = \{\zeta_i, (1 \wedge 2) \rightsquigarrow \langle\pi_i\rangle\zeta_i, 1 \leq i \leq n\}$.

Since 1 is a basic FS, $\pi_i^1 = \emptyset$, or $\pi_i^1 = x_i^1$, and $x_i^1 \sqsubseteq \psi_i^1$ by the definition of TDFSs and their translation into $\mathcal{L}_{pdu}$, and the axiom REV. Similarly for 2.

So

$$\begin{aligned} \pi^{12} = \; & \cup_{1 \leq i \leq n} \pi_i^1 \cup \pi_i^2 \\ \subseteq \; & \{\langle x_i^1, t_1\rangle, \langle x_i^2, t_2\rangle : 1 \leq i \leq n\} \end{aligned}$$

Where t_1 and t_2 are the root types of 1 and 2. So $Spec^{12}_{\pi} \sqcap Comp^{12}_{\pi} \sqsupseteq \sqcap\phi$. Furthermore, by TU1, TU2 and REV, $\psi \sqsupseteq \sqcap\{\psi_i^1, \psi_i^2 : 1 \leq i \leq n\} \sqsupseteq \sqcap\phi$. So $Spec^{12}_{\pi} \sqcap Comp^{12}_{\pi} \sqcap \psi \sqsupseteq \sqcap\phi$. That is, $\delta \sqsupseteq \sqcap\phi$. And since $\sqcap\phi \neq \bot$, $\delta \neq \bot$. So by theorem 2 (The Typed Unification Property), $\delta = \sqcap\phi$. $\Box$

5.6 Different Tails for Different Sublanguages

If we restrict the kinds of TDFSs that are permitted, then we can reduce the length of tails, without sacrificing the properties we have proved for $\overset{<>}{\sqcap}$ — namely, order independence and reduction to monotonic typed unification where there are no conflicts. There is a balance between restricting the language of TDFSs, and the potential length of tails. The more restricted the language, the less of the unification history needs to be recorded in order to retain order independence. The computational complexity of $\overset{<>}{\sqcap}$ is $O(nt^2)$ where n is the number of nodes in the TDFSs and t is the length of the tails, so reducing the length of the tails would bring $\overset{<>}{\sqcap}$ closer in actual complexity to monotonic typed unification.

First, consider the following sublanguage: suppose that whenever $t_1 \sqsubset t_2$, then for any TDFSs of the forms below, either $a \sqcap b = \bot$, or $a \sqsubseteq b$.

$$\begin{bmatrix}\mathbf{t}_1 \\ \mathrm{F} : /\mathbf{a}\end{bmatrix} \text{ and } \begin{bmatrix}\mathbf{t}_2 \\ \mathrm{F} : /\mathbf{b}\end{bmatrix}$$

With this sublanguage, we can reduce the amount of information in tails, and yet the existing axioms will still produce an order independent $\overset{<>}{\sqcap}$ which reduces to typed unification where there is no conflict. The definition of tails for the base case remains the same. The revised definition of how to construct new tails from old ones is given below:

$$\pi^{12} = \cup^{\pi'}_{(1\wedge 2)>\langle\pi\rangle\approx\langle\pi'\rangle}\{\langle a,t\rangle : \langle a,t\rangle \in \pi'^1 \cup \pi'^2 \text{ and } \forall\langle a',t'\rangle \in \cup^{\pi'}(\pi'^1 \cup \pi'^2), t' \not\sqsubset t\}$$

Now, π^{12} is not constructed by set union on π^1 and π^2, and all the tails reentrant with π. Rather, it just retains the values associated with the most specific types. Indeed, π^{12} is potentially shorter than π^1 or π^2. For example, if $\pi^1 = \{\langle a,t_1\rangle, \langle b,t_2\rangle\}$, $\pi^2 = \{\langle c,t_3\rangle\}$, where $t_3 \sqsubset t_2$ and $t_3 \sqsubset t_1$, and there are no reentrancies with π, then $\pi^{12} = \{\langle c,t_3\rangle\}$. Whereas under the old definition, π^{12} was $\{\langle a,t_1\rangle, \langle b,t_2\rangle, \langle c,t_3\rangle\}$.

With this new definition of tails and the existing axioms of the logic, all the above lemmas and theorems hold, but only for the restricted language.[13] In fact, this new definition of tails is analogous to the strategy for defining $\overset{<>}{\sqcap}$ that we mentioned in §3.2, which involved reasoning about a hierarchy of indices, which recorded the specificity from which default paths originated. The new definition of tails, like the index strategy, records essentially only the most specific defaults, and doesn't keep track of overridden values. Recording this amount of information from the unification history is sufficient for order independence on this restricted sublanguage.

This sublanguage might be appropriate for some linguistic applications: for example, it is sufficient for defining the type constraints given in Figure 7. The problem is the difficulty in ensuring that the extra condition holds. In general, it is highly desirable for usability of a formalism that any linguistic description can be checked efficiently to ensure that it is well-formed according to the language. But even if we assume that defaults are always introduced by the type hierarchy, and check that the extra sublanguage condition holds of that, this does not ensure that it will hold of any possible unification. For example, suppose that t_1 and t_2 have the following constraints (which do meet the condition):

$$\begin{bmatrix}\mathbf{t}_1 \\ \mathrm{F} : /\mathbf{a}\end{bmatrix} \quad \begin{bmatrix}\mathbf{t}_2 \\ \mathrm{F} : /\mathbf{b}\end{bmatrix}$$

where $t1 \sqsubset t2$, $a \sqsubset b$

If we also have c such that $b \sqcap c = d$ and $d \sqcap a = e$ then unifying a TDFS of type t_2 with the strict information that the value of F is c gives us the following TDFSs which would not meet the condition for the sublanguage if unified:

[13] The proof is left as an exercise for the reader.

$\begin{bmatrix} \mathbf{t}_1 \\ \mathrm{F} : /\mathbf{a} \end{bmatrix} \quad \begin{bmatrix} \mathbf{t}_2 \\ \mathrm{F} : \mathbf{c}/\mathbf{d} \end{bmatrix}$

$d \sqcap a \neq \bot$, $a \not\sqsubseteq d$

Other examples involving default information can be devised. A more stringent but statically checkable condition would be to require all specified default values to be maximally specific. This might be suitable for some applications — it is still adequate for the example in Figure 7 for instance — but it is clearly significantly less expressive.

We now consider a version of $\overset{<>}{\sqcap}$ for the sublanguage used in Young and Rounds (1993), assuming that multiple solutions are treated as an error condition. That is, we remove reentrancies, types, and the subsumption relation on types. Then we define tails as follows: For the base case:

$$\pi^1 = \begin{cases} \{a\} & \text{If } 1 \Rightarrow \langle\pi\rangle a \text{ and } 1 \not\rightsquigarrow \langle\pi\rangle a \\ \emptyset & \text{otherwise} \end{cases}$$

We build tails as follows. Suppose $(1 \wedge 2) \rightsquigarrow \langle\pi\rangle\psi$. Then:

$$\pi^{12} = \begin{cases} \pi^1 \sqcap \pi^2 & \text{If } \pi^1 \sqcap \psi \neq \bot \text{ and } \pi^2 \sqcap \psi \neq \bot \\ \pi^1 & \text{If } \pi^1 \sqcap \psi \neq \bot \text{ and } \pi^2 \sqcap \psi = \bot \\ \pi^2 & \text{If } \pi^1 \sqcap \psi = \bot \text{ and } \pi^2 \sqcap \psi \neq \bot \\ \emptyset & \text{If } \pi^1 \sqcap \psi = \bot \text{ and } \pi^2 \sqcap \psi = \bot \end{cases}$$

And we define:

$$Spec^1_\pi = Comp^1_\pi = \pi^1$$

Now, all tails are empty or singletons, and hence in general they are smaller than they were for the general language. But this new definition of tails still produces an order independent version of $\overset{<>}{\sqcap}$. Obviously now, we have to change the axioms slightly, because default reentrancies must be removed. So we remove RE2, and modify CON and PDU to the more restricted versions given below:

CON `Consistency`

$$\frac{\begin{array}{l} 1 \Rightarrow \langle\pi\rangle a \\ (1 \rightarrow \langle\pi\rangle \approx \langle\pi'\rangle) \rightarrow 2 \not\Rightarrow \langle\pi'\rangle\top \\ (1 \wedge 2) \not\rightarrow \langle\pi_1\pi_2\rangle \approx \langle\pi_2\rangle \end{array}}{(1 \wedge 2) \Rightarrow \langle\pi\rangle a}$$

PDU `The PDU Axiom`

$$\frac{\begin{array}{l} 1 \Rightarrow \langle\pi\rangle a \\ 2 \Rightarrow \langle\pi'\rangle b \\ (1 \rightarrow \langle\pi\rangle \approx \langle\pi'\rangle) \\ (1 \wedge 2) \not\rightarrow \langle\pi_1\pi_2\rangle \approx \langle\pi_2\rangle \end{array}}{(1 \wedge 2) \Rightarrow \langle\pi\rangle(Spec^{12}_\pi \sqcap Comp^{12}_\pi \sqcap \psi)}$$

Where:

$\psi = \sqcap\{\psi_{\pi_i} : 1 \wedge 2 \rightsquigarrow \langle\pi_i\rangle\psi_{\pi_i} \text{and } (1 \wedge 2) \rightarrow \langle\pi\rangle \approx \langle\pi_i\rangle\}$

This version of $\overset{<>}{\sqcap}$ is equivalent to that given in Young and Rounds (1993), assuming that multiple extensions are treated as an error condition. So their version of persistent default unification is a special case of the operation defined in §3.5. It can be implemented more efficiently than $\overset{<>}{\sqcap}$, but the lack of default reentrancy and subsumption relationships on the types mean that it is insufficiently expressive to model some of the phenomena we are interested in.

6 Linguistic Applications

Our illustrative lexical hierarchy concerning English inflectional morphology was chosen because it is the simplest example capable of demonstrating the inadequacies of existing definitions of default unification and exemplifying the properties of persistent default unification of TDFSs. In this section, we discuss further applications which we think provide better motivation for the introduction of such an operation into constraint-based linguistic theories.

There are many papers which argue for default inheritance of most types of lexical property ranging from cases of inflectional morphology of the type discussed above, through processes of derivation, conversion and sense extension (see Daelemans *et al.*, 1992; Briscoe, 1993 for reviews). In addition, default inheritance has been argued to play a role in feature propagation for syntactic description (Gazdar *et al.*, 1985; Shieber, 1986b), in the specification of relations between constructions (Goldberg, 1992). The applications we briefly discuss involve treatments of paradigmatic relations between lexical items and between constructions, processes of regular sense extension and syntactic feature propagation.

6.1 Lexical Semantic Specifications

Briscoe *et al.* (1990), Boguraev and Pustejovsky (1990, 1993) and Copestake and Briscoe (1992) argue that formalisation of Pustejovsky's (e.g. 1991) approach to lexical semantics requires a notion of default inheritance in the lexicon. For instance, Pustejovsky argues on linguistic grounds that nouns denoting artifacts must incorporate a representation of their telic role or 'purpose' in order to account for the interpretation of 'logically' metonymic constructions such as (18).

(18) a. John began a new book
b. Jane finished her beer
c. Bill enjoyed the film
d. Mary likes hamburgers

In each case the noun phrase object denotes an artifact and the (usual) interpretation of the verb phrase involves a 'coercion' of the artifact to a process or event involving the purpose of that artifact – reading books, drinking beer, watching films and eating hamburgers. Pustejovsky emphasises that the assumption that certain verbs cause such coercions has direct linguistic consequences for the insightful representation of argument structure and complementation; for example, *enjoy* can subcategorise for a complement denoting a process in which the experiencer subject of *enjoy* participates, capturing the relatedness in meaning between (19a) and (19b).

(19) a. John enjoyed the play
b. John enjoyed watching the play

Once we accept the utility of representing telic roles, it is clear that they are a natural candidate for an inheritance based treatment; for example, the telic role of liquid artifacts

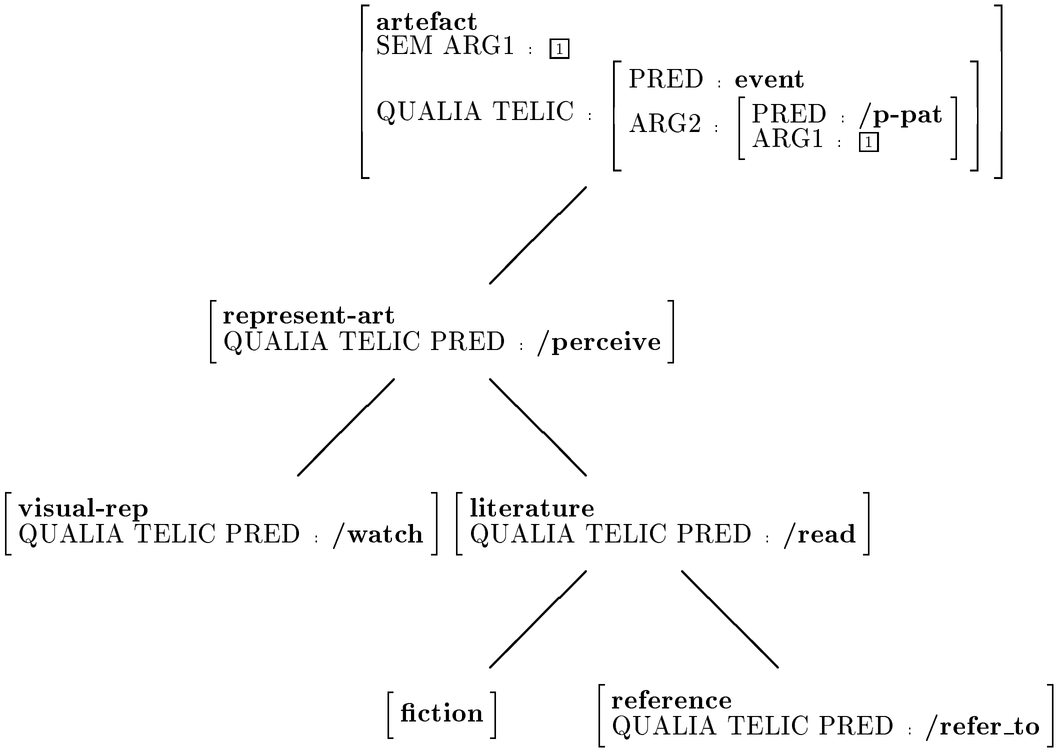


Figure 16: Inheritance of Telic Roles

will be *drink*, that of visual representation artifacts *watch*, and so forth. It is also not difficult to show that inheritance must be default; for example, the telic role of 'literature' will be *read*, however that of the subclass of reference books will be *refer-to*. Briscoe *et al.* propose to express these observations in a unification-based account by representing the lexicon in terms of overwriting templates (e.g. Karttunen, 1986), and Copestake (1992, 1993) proposes using a non-associative version of default unification. Boguraev and Pustejovsky (1990, 1993), Copestake and Briscoe (1992, 1994) and Vossen and Copestake (1993) discuss the extension of the technique to the multiple orthogonal default inheritance of nominal qualia structure (e.g. Pustejovsky, 1991), in general. In the current framework these proposals can be incorporated directly without loss of declarativity, using the type of lexical hierarchy introduced in Figure 7. The value of the TELIC attribute will be a complex FS representing the semantics of specified (generic) verbs with the artifact associated (usually) with the object argument slot. A fragment of such a hierarchy is shown in Figure 16. The coercion process itself can be implemented as a unary rule (Briscoe *et al.*1990) or by monotonic type constraints (Copestake and Briscoe, 1994).

It is clear that the examples in (18) only have defeasible interpretations involving the 'usual purpose' readings. In suitable discourse contexts, these readings can be replaced by ones involving more specific processes, different qualia roles, and so forth, as (20) illustrates.

(20) a. Coppola particularly enjoyed that film (watching $\rightsquigarrow$ making)

b. You can see our neighbour, 'Howard Hughes', really enjoying his new car every Sunday in the drive, hose in hand (driving $\rightsquigarrow$ cleaning)

c. Fido obviously enjoyed my new book (reading $\rightsquigarrow$ eating)

Briscoe *et al.* (1990) suggest that this can be accommodated by making the rule of coercion which converts an object denoting NP into a process or event denoting NP create a default specification of this process or event, based on the telic role of the noun (if available). In the present context of TDFSs, this proposal makes more formal sense since it reduces to making this aspect of the FS associated with the NP defeasible. Thus, in contrast with the morphology example shown in Figure 7, we do not assume that the default structure is incorporated into the non-defeasible structure before non-lexical processing occurs.

The semantic translation we have provided for TDFSs interprets a defeasible specification such as *event/write* as a default conditional, providing the foundation for an interface with a theory of discourse interpretation based on a default conditional logic (e.g. Lascarides and Asher, 1993). Briscoe *et al.* (1990) argue on the basis of corpus data that the notion of a lexically specified default interpretation is motivated by the distribution of default and non-default interpretations with verbs such as *enjoy* which make available both an explicit (progressive VP) and implicit (NP) complementation pattern for realising the intended meaning. Since the implicit pattern is chosen mostly when the default interpretation is correct or the discourse context is informationally rich, conflicting and determinate, whilst the explicit pattern is chosen mostly when the default reading would be inappropriate and the context is not determinate, this suggests that language users structure their utterances in a manner which supports assumption of the default reading in the absence of conflicting information, but rejection of it when additional conflicting information is made available. This view of the interaction of semantic specification with discourse interpretation is fully compatible with the principles of the DICE framework (e.g. Lascarides and Asher, 1993, Asher and Lascarides, 1994).

Lascarides (1995) shows how the defeasible pragmatic information in DICE and defeasible results in PDU can be made to communicate in a perspicuous fashion, via just two DICE axioms. The details are beyond the scope of this paper, because it would require a detailed discussion of the DICE framework. However, we informally describe this approach. The two axioms predict when the PDU default predictions survive in the discourse context and when they are overridden. They also guarantee that the PDU indefeasible results *always* survive, regardless of what pragmatic information is stated. The two axioms make essential use, therefore, of the fact that defeasible results of PDU are 'visibly' marked as such for the pragmatic component. Furthermore, we would argue that default lexical specification could not be replaced in this instance with, for example, a disjunction of alternative values to be selected from pragmatically because of the open-ended nature of the eventual interpretation. Similarly, we would argue that not providing a default lexical specification and leaving the coercion predicate underspecified is not compatible with the corpus-based evidence summarised above from Briscoe *et al.* (1990), since this would require the pragmatic component to provide a default interpretation which is specified lexically and not in terms of contextual information.

The first axiom states that normally, default PDU results survive in the discourse context. The second axiom ensures that when there is discourse information that conflicts with the defeasible results of PDU, the discourse information wins, even if it's defeasible. In the case of metonymy, Lascarides (1995) demonstrates how the axioms predict different interpretations of (21a) and (21b).

(21) a. John enjoyed the book.
b. The goat enjoyed the book.

(21a) is interpreted as John enjoyed reading the book, because the PDU expansion of the metonymy survives in this context. In contrast, this PDU expansion of metonymy is blocked by the second axiom in the interpretation of (21b), because of the conflicting defeasible domain information that goats don't read. We refer the reader to Lascarides (1995) for further details.

To summarise, we have sketched treatment of default inheritance of lexical qualia roles which we argue improve on those presented in Briscoe *et al.*(1990) in that they are declarative and can persist beyond the lexicon, and can thus interact straightforwardly with a theory of discourse interpretation grounded in default logic.

6.2 Semantic Broadening

Copestake and Briscoe (1994) argue for a systematic process of sense modulation which they term *broadening* where sense usages are available in context which appear to semantically subsume the basic sense of a lexeme/sign. Usually it appears that a quale, in Pustejovsky's (1991) sense, which is specified in the basic sense becomes overridden in context. For example, the normal usages of *bank* and *cloud* could be specified as stating both form and composition (earth/water vapour). However, both have usages where alternative compositions are stated *bank of rhododendrons*, *bank of clouds/cloud bank*, *cloud of mosquitoes*, *dust cloud*. In some comparable cases the broadened sense may appear more metaphorical, for example *forest of hands*. In many cases there is evidence that broadening of meaning has taken place diachronically and that the original senses tended to be specific and concrete (see Sweetser 1990). It seems appropriate to regard these examples in terms of a modulation of sense rather than a complete shift, but this modulation is most naturally expressed as being non-monotonic. There is a very strong preference for one particular sense and the alternative interpretations are not conventionalised, but given by context (there is no conventional interpretation of *cloud* as *cloud of mosquitoes*). This implies that non-default interpretations will only be usual in contexts which explicitly give the exceptional component (normally by compounding or post-modification).

To represent broadening we make use of lexically specified persistent default components of the qualia structure and allow these to be overridden. In the FS for the lexical entry for *cloud* shown in Figure 17 the qualia structure is stated to refer necessarily to an individuated physical object of amorphous form, with a composition that is also physical and refers cumulatively (i.e. the composition is either a mass or a plural object). By default, *cloud* is a natural object (as opposed to an artifact) and is composed of water vapour.[14] Referring to the process of overriding the lexically specified defaults as broadening is perhaps somewhat

[14] This description has been somewhat simplified but in any case we would not claim that it is completely adequate. It does not, for instance, cover the mass use of *cloud*, found in (22a), which seems to be available only with the default usage (compare (22b)):

(22) a. We flew into dense cloud.
b. * We walked into dense cloud of smoke.

Nor does it cover the metaphorical uses, such as *cloud of suspicion*.

$$
\left[\begin{array}{l}
\textbf{lex-count-noun} \\
\text{ORTH} : \textbf{cloud} \\
\text{CAT} : \textbf{noun-cat} \\
\text{SEM} : \textbf{obj-noun-formula} \\
\text{QUALIA} : \left[\begin{array}{l}
\textbf{phys\_obj} \ / \ \textbf{natural\_obj} \\
\text{FORM} : \left[\begin{array}{l}
\textbf{nomform} \\
\text{RELATIVE} : \textbf{individuated} \\
\text{ABSOLUTE} : \textbf{amorphous}
\end{array}\right] \\
\text{CONSTITUENCY} : \textbf{phys\_cum} \ / \ \textbf{water-vapour}
\end{array}\right]
\end{array}\right]
$$

Figure 17: Lexical entry for *cloud*

misleading, since a more general FS never actually exists in isolation according to this treatment. The intuition that the sense is broadened is reflected in the non-defeasible components of the modified structure, however: for example the semantic contribution of *cloud* to *cloud of mosquitoes* could be represented as a FS with unspecified composition.

Broadening and other kinds of sense modulation (Cruse, 1986) provide a straightforward motivation for the persistence of default lexical specifications into the syntagmatic plane. Persistent default unification provides a natural mechanism for implementing interactions of this kind between lexical semantic specification and syntactic and compositional semantic rules. The axioms specified in Lascarides (1995) for predicting when defeasible results of $\overset{<>}{\sqcap}$ ultimately survive into the interpretation would account for semantic broadening, as well as the expansion of metonymy described earlier. Alternative treatments, such as alternate entries for *cloud* and *bank* which specify an indefeasible value for composition in the absence of a complement and leave it underspecified with obligatory complementation seem unsatisfactory since this would predict that in a narrative such as (23) the second occurrence of *cloud* denotes a mass composed of water vapour.

(23) The cow was engulfed by a cloud of flies, flicking its tail ineffectually. As the cloud dissipated, we realised it was injured.

6.3 Dative Constructions

Goldberg (1992), building on the frameworks of frame semantics and construction grammar developed by Fillmore and his colleagues (e.g. Fillmore *et al.*, 1988), argues that certain constructions should be seen as related by 'metaphorical or polysemous links'. For example, she argues that there is a family of dative constructions which exhibit the same syntactic properties and related semantic properties, exemplified in (24).

(24)
- a. Mary gave Joe a present
- b. Joe painted Sally a picture
- c. The medicine brought him relief
- d. The music lent the party a festive air
- e. Jo gave Bob a punch
- f. He blew his wife a kiss

Goldberg argues that the core ditransitive construction involves a volitional agent and willing recipient and carries the entailments that the agent causes the recipient to receive the object denoted by the patient argument, as in (24a). Under this interpretation, (24b) involves a shift in meaning where the recipient may or may not receive the patient, but the agent acts with this intention. Following Sanfilippo (1990, 1993) we might represent these differences in the basic (abstract) meaning of the construction in terms of entailments associated with proto thematic roles (Dowty, 1989), so that 'agent' becomes *p-agt-cause-transfer* in (24a) and *p-agt-cause-make-intend-transfer* in (24b). In the case of these first two examples it is plausible to argue that there are lexical rules which relate the dative construction with the alternative complementation patterns involving *to* and *for* PP arguments, respectively, and that this rule alters the proto-agent role of a 'creation' verb such as *paint* from *p-agt-cause-make* to *p-agt-cause-make-intend-transfer*. Furthermore, we will represent the different entailments (willingness and successful transfer) concerning the first object in (24a) and (24b) as 'recipient' and 'benefactive', respectively.

As Goldberg argues, if we want to elegantly capture the similarity between the dative constructions in these two rules it should only be necessary to state the form of the construction once. Furthermore, it should not be necessary to say that verbs of creation, such as *paint* are lexically ambiguous between two and three -place predicates; rather it is participation in the dative construction itself which creates a third beneficiary argument for these inherently two-place predicates. In what follows, we assume a lexically-based theory of grammar in which bounded lexically-governed dependencies of the type discussed by Goldberg are treated in terms of the subcategorisation requirements of their heads (in this case verbs). We also assume an approach to lexical rules, in which such rules conditionally create derived lexical signs stated in terms of mappings between (lexical) types (e.g. Pollard and Sag, 1987), and factor the majority of information concerning the form of the types related into their associated type constraints (e.g. Flickinger and Nerbonne, 1992). Thus, the core dative rule and benefactive dative rule could be represented as in Figures 18 and 19, assuming the (inherited and specific) type constraints on 'transfer' ditransitive and 'creation' transitive verbs and on the dative construction (sign) given in Figures 20, 21 and 22.[15]

As **benef-dative-verb** is a subtype of **dative-sign**, the benefactive dative lexical rule will create a new lexical entry for 'creation' verbs which inherits properties of the dative construction. However, **benef-dative-verb** overrides the specific default thematic entailments associated with the basic construction. Thus the resulting sign will have the proto-roles **p-agt-cause-make-intend-transfer** and **p-pat-benef**, rather than **p-agt-cause-transfer** and **p-pat-recip**.[16] On the other hand, when the core (recipient) dative lexical rule is applied to a sign of type **lit-transfer-ditrans-ppto-verb**, the specifications of the proto-roles as **p-agt-cause-transfer** and **p-pat-recip** in the result will be indefeasible, in contrast to their defeasible status in **dative-sign**. This is because they are indefeasible in the type **lit-**

[15]In this example and those that follow, we make as few assumptions as possible concerning the exact grammatical framework employed. The basic ideas are compatible with HPSG, LFG or UCG -like treatments of the realisation of grammatical relations, and the proto-role approach to thematic relations might be linked to an event-based semantic framework (e.g. Parsons, 1990; Sanfilippo, 1990, 1993) or to a LFG style treatment in terms of mappings or projections to a semantic representation (Halvorsen and Kaplan, 1988; Dalrymple *et al.*, 1993). We assume that proto-role entailments create selection preferences on the semantic nature of their arguments, but do not attempt to make these constraints explicit. Similarly, we make no attempt to show how the generalisations implicit in type constraints can be elegantly factored out into constraints on supertypes (e.g. Pollard and Sag, 1987; Sanfilippo, 1993).

[16]Note that although for the sake of clarity we show the OBJ1 and the **p-pat-benef** role in Figure 19, this information is inherited from the **benef-dative-verb** subtype of the dative construction and not stipulated directly in the rule.

$$\left[\begin{array}{l} \textbf{transfer-ditrans-ppto-verb} \\ \text{PHON} : \left[\text{STEM} : \boxed{1}\right] \\ \text{SYN} : \left[\begin{array}{l} \text{SUBJ} : \left[\text{SEM} : \boxed{2}\right] \\ \text{OBJ1} : \left[\text{SEM} : \boxed{3}\right] \\ \text{OBL1} : \left[\text{SEM} : \boxed{4}\right] \end{array}\right] \\ \text{SEM} : \boxed{5} \end{array}\right] \mapsto \left[\begin{array}{l} \textbf{recip-dative-verb} \\ \text{PHON} : \left[\text{STEM} : \boxed{1}\right] \\ \text{SYN} : \left[\begin{array}{l} \text{SUBJ} : \left[\text{SEM} : \boxed{2}\right] \\ \text{OBJ1} : \left[\text{SEM} : \boxed{4}\right] \\ \text{OBJ2} : \left[\text{SEM} : \boxed{3}\right] \end{array}\right] \\ \text{SEM} : \boxed{5} \end{array}\right]$$

$$\textbf{recip-dative-verb} \sqsubset \textbf{dative-sign}$$

Figure 18: Dative Lexical Rule

$$\left[\begin{array}{l} \textbf{create-dyadic-trans-verb} \\ \text{PHON} : \left[\text{STEM} : \boxed{1}\right] \\ \text{SYN} : \left[\begin{array}{l} \text{SUBJ} : \left[\text{SEM} : \boxed{2}\right] \\ \text{OBJ1} : \left[\text{SEM} : \boxed{3}\right] \end{array}\right] \\ \text{SEM} : \left[\text{PRED} : \boxed{4}\right] \end{array}\right]$$

$$\mapsto \left[\begin{array}{l} \textbf{benef-dative-verb} \\ \text{PHON} : \left[\text{STEM} : \boxed{1}\right] \\ \text{SYN} : \left[\begin{array}{l} \text{SUBJ} : \left[\text{SEM} : \boxed{2}\right] \\ \text{OBJ1} : \left[\text{SEM} : \boxed{5}\right] \\ \text{OBJ2} : \left[\text{SEM} : \boxed{3}\right] \end{array}\right] \\ \text{SEM} : \left[\begin{array}{l} \text{PRED} : \boxed{4} \\ \text{ARG1} : \left[\begin{array}{l} \text{PRED} : \textbf{p-agt-cause-make-intend-transfer} \\ \text{ARG} : \boxed{2} \end{array}\right] \\ \text{ARG2} : \left[\begin{array}{l} \text{PRED} : \textbf{p-pat-benef} \\ \text{ARG} : \boxed{5} \end{array}\right] \\ \text{ARG3} : \left[\begin{array}{l} \text{PRED} : \textbf{p-pat-aff-obj} \\ \text{ARG} : \boxed{3} \end{array}\right] \end{array}\right] \end{array}\right]$$

$$\textbf{benef-dative-verb} \sqsubset \textbf{dative-sign}$$

Figure 19: Benefactive Dative Lexical Rule

$$\left[\begin{array}{l}
\textbf{lit-transfer-ditrans-ppto-verb} \\
\text{PHON} : \left[\text{STEM} : \textbf{stem}\right] \\
\text{SYN} : \left[\begin{array}{l}
\text{CAT} : \textbf{verb} \\
\text{SUBJ} : \left[\begin{array}{l} \text{SYN} : \left[\text{CAT} : \textbf{np}\right] \\ \text{SEM} : \boxed{2} \end{array}\right] \\
\text{OBJ1} : \left[\begin{array}{l} \text{SYN} : \left[\text{CAT} : \textbf{np}\right] \\ \text{SEM} : \boxed{3} \end{array}\right] \\
\text{OBL1} : \left[\begin{array}{l} \text{SYN} : \left[\text{CAT} : \textbf{pp}\right] \\ \text{SEM} : \boxed{4} \end{array}\right]
\end{array}\right] \\
\text{SEM} : \left[\begin{array}{l}
\text{PRED} : \textbf{pred} \\
\text{ARG1} : \left[\begin{array}{l} \text{PRED} : \textbf{p-agt-cause-transfer} \\ \text{ARG} : \boxed{2} \end{array}\right] \\
\text{ARG2} : \left[\begin{array}{l} \text{PRED} : \textbf{p-pat-recip} \\ \text{ARG} : \boxed{4} \end{array}\right] \\
\text{ARG3} : \left[\begin{array}{l} \text{PRED} : \textbf{p-pat-aff-obj} \\ \text{ARG} : \boxed{3} \end{array}\right]
\end{array}\right]
\end{array}\right]$$

$$\textbf{lit-transfer-ditrans-ppto-verb} \sqsubset \textbf{transfer-ditrans-ppto-verb}$$

Figure 20: Ditransitive Verb Type Constraint

$$\left[\begin{array}{l}
\textbf{create-dyadic-trans-verb} \\
\text{PHON} : \left[\text{STEM} : \textbf{stem}\right] \\
\text{SYN} : \left[\begin{array}{l}
\text{CAT} : \textbf{verb} \\
\text{SUBJ} : \left[\begin{array}{l} \text{SYN} : \left[\text{CAT} : \textbf{np}\right] \\ \text{SEM} : \boxed{2} \end{array}\right] \\
\text{OBJ1} : \left[\begin{array}{l} \text{SYN} : \left[\text{CAT} : \textbf{np}\right] \\ \text{SEM} : \boxed{3} \end{array}\right]
\end{array}\right] \\
\text{SEM} : \left[\begin{array}{l}
\text{PRED} : \textbf{pred} \\
\text{ARG1} : \left[\begin{array}{l} \text{PRED} : \textbf{p-agt-cause-make} \\ \text{ARG} : \boxed{2} \end{array}\right] \\
\text{ARG2} : \left[\begin{array}{l} \text{PRED} : \textbf{p-pat-aff-obj} \\ \text{ARG} : \boxed{3} \end{array}\right]
\end{array}\right]
\end{array}\right]$$

Figure 21: Transitive Type Constraint

$$
\left[\begin{array}{ll}
\textbf{dative-sign} & \\
\text{PHON} : & \left[\text{STEM} : \textbf{stem}\right] \\
\text{SYN} : & \left[\begin{array}{ll}
\text{CAT} : \textbf{verb} & \\
\text{SUBJ} : & \left[\begin{array}{l} \text{SYN} : \left[\text{CAT} : \textbf{np}\right] \\ \text{SEM} : \boxed{2} \end{array}\right] \\
\text{OBJ1} : & \left[\begin{array}{l} \text{SYN} : \left[\text{CAT} : \textbf{np}\right] \\ \text{SEM} : \boxed{3} \end{array}\right] \\
\text{OBJ2} : & \left[\begin{array}{l} \text{SYN} : \left[\text{CAT} : \textbf{np}\right] \\ \text{SEM} : \boxed{4} \end{array}\right]
\end{array}\right] \\
\text{SEM} : & \left[\begin{array}{ll}
\text{PRED} : \textbf{pred} & \\
\text{ARG1} : & \left[\begin{array}{l} \text{PRED} : \textbf{p-agt/p-agt-cause-transfer} \\ \text{ARG} : \boxed{2} \end{array}\right] \\
\text{ARG2} : & \left[\begin{array}{l} \text{PRED} : \textbf{p-pat/p-pat-recip} \\ \text{ARG} : \boxed{3} \end{array}\right] \\
\text{ARG3} : & \left[\begin{array}{l} \text{PRED} : \textbf{p-pat-aff-obj} \\ \text{ARG} : \boxed{4} \end{array}\right]
\end{array}\right]
\end{array}\right]
$$

Figure 22: Dative Type Constraint

transfer-ditrans-ppto-verb. The lexical rule also specifies that the entire semantics of the derived sign is reentrant with that of the basic sign. These rules predict the correct relationships between (24a) and (24b), and the corresponding sentences with *to* and *for* complements.

Further support for this approach to the dative construction and dative lexical rule(s) is provided by the other examples in (24). Goldberg argues that (24c,d) are licensed by a metaphorical extension of the transfer relation by which causal events are viewed as transfers. Causing an effect in an entity is understood as transferring that effect to it. We might capture this by altering the entailments associated with verbs such as *lend* by the proto-roles specified by **lit-transfer-ditrans-ppto-verb** by a lexical rule which created an entry of a sister type **met-transfer-ditrans-ppto-verb** which specified different proto-roles. The dative lexical rule would also apply to this subtype, capturing the fact that this metaphorical extension in the dative construction parallels a similar extension of the same verb set in the oblique *to* prepositional phrase construction. Finally, (24e,f) provide evidence that certain quasi-idiomatic expressions need to be associated directly with **dative-sign** as they have no counterparts in such oblique expressions. We assume that (quasi-)idioms are best represented as subtypes of lexical signs in which not only the syntactic head but also other arguments are severely constrained in terms of lexical selection. Thus Goldberg claims the quasi-idiomatic expressions in (24e,f) are licensed by a metaphor which involves understanding actions intentionally directed at another person as being entities transferred to that person. As a first approximation, we might represent this process in terms of the subtype of **dative-sign** shown in Figure 23, in which we have overridden the default proto-role specifications with entailments specific to the metaphor which we assume also serve to constrain the range of acceptable arguments to the (transfer) verb.

To summarise, we have sketched a lexical treatment of dative constructions intended to capture and formalise some of Goldberg's observations concerning similarities and differences

$$
\begin{bmatrix}
\textbf{met-dative-sign} & \\
\text{PHON} : & \begin{bmatrix} \text{STEM} : \textbf{stem} \end{bmatrix} \\
\text{SYN} : & \begin{bmatrix}
\text{CAT} : \textbf{verb} & \\
\text{SUBJ} : & \begin{bmatrix} \text{SYN} : \begin{bmatrix} \text{CAT} : \textbf{np} \end{bmatrix} \\ \text{SEM} : \boxed{2} \end{bmatrix} \\
\text{OBJ1} : & \begin{bmatrix} \text{SYN} : \begin{bmatrix} \text{CAT} : \textbf{np} \end{bmatrix} \\ \text{SEM} : \boxed{3} \end{bmatrix} \\
\text{OBJ2} : & \begin{bmatrix} \text{SYN} : \begin{bmatrix} \text{CAT} : \textbf{np} \end{bmatrix} \\ \text{SEM} : \boxed{4} \end{bmatrix}
\end{bmatrix} \\
\text{SEM} : & \begin{bmatrix}
\text{PRED} : \textbf{pred} & \\
\text{ARG1} : & \begin{bmatrix} \text{PRED} : \textbf{p-agt-cause-met-transfer} \\ \text{ARG} : \boxed{2} \end{bmatrix} \\
\text{ARG2} : & \begin{bmatrix} \text{PRED} : \textbf{p-pat-met-recip} \\ \text{ARG} : \boxed{3} \end{bmatrix} \\
\text{ARG3} : & \begin{bmatrix} \text{PRED} : \textbf{p-pat-transferred-ev} \\ \text{ARG} : \boxed{4} \end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

Figure 23: Metaphorical Dative Type Constraint

between related versions of the same basic construction. We have utilised a familiar and standard notion of lexical rule, coupled with persistent default unification of TDFSs. In this formulation of this fragment of data, there is no requirement for the default specifications to persist outside the lexicon. However, a more adequate treatment of proto-roles in this framework would probably involve persistent default lexical specification of the entailments associated with proto-roles which could be overridden at the compositional semantic or pragmatic level.

6.4 Syntactic Feature Propagation

Shieber (1986b) and Bouma (1992) discuss the unification-based interpretation of the Head Feature Convention (HFC) of GPSG (Gazdar *et al.*, 1985). The HFC is a default convention which equates the values of head features on the mother and head daughter in a local tree unless these features have acquired distinct values through stipulation in immediate dominance rules. Most unification-based approaches to GPSG either captured the default nature of the HFC and related feature propagation principles through a non-unification-based metagrammatical pre-compilation phase (e.g. Briscoe *et al.*, 1987) or simplified it to a non-default reentrancy condition (Shieber, 1986a,b). Bouma (1992) presents a version of default unification which he motivates as a technique for implementing the HFC in a unification-based fashion. Bouma's version of default unification is an asymmetric, non-associative operation which adds consistent information from a defeasible FS to an indefeasible FS. Bouma represents the HFC as a default reentrancy associated with each underspecified rule, schematically of the form in (25a), in which mother and head daughter categories are FSs and the head daughter FS is taken to be indefeasible.

(25) a. $M(other) \rightarrow X\ H(ead)D(aughter)\ Y$

b. $S \rightarrow NP\ H[BAR1]$

c. $[N-, V+, BAR2] \rightarrow NP\ [BAR1]$

d. $[N-, V+, BAR2] \rightarrow NP\ [N-, V+, BAR1]$

The idea is that default unification can be used in a pre-compilation phase to flesh out underspecified rules, so that a rule such as (25b) which contains the features in (25c) will be expanded out to (25d), where BAR values remain distinct because of the stipulation on the head daughter, though BAR is a head feature. After compilation the expanded rule can be utilised for parsing using monotonic unification. As rule expansion serves only to create a set of fully specified (phrase-structure) rules, the overall approach remains declarative because default unifications will never be 'chained'.

A weakness of Bouma's approach is that it preserves the declarativity of the overall grammar only by making strong assumptions concerning the point at which such feature propagation principles are applied (which are also at odds with those assumed in GPSG). If, for example, it turned out that it was more efficient to apply such unification-based constraints after the initial construction of (underspecified) syntactic descriptions during parsing (see e.g. Maxwell and Kaplan (1993) for evidence that this is true for some types of grammar), then different results might be obtained depending on whether the sequence of default unifications required to flesh out a chain of mothers and head daughters in a syntactic description was implemented top-down or bottom-up.

We might be able to utilise persistent default unification of TDFSs in order to formalise the HFC by making use of the flexibility in the definition of specificity mentioned in §4.2 and calculate the inheritance relationship between head features on the basis of the relationship given by the syntactic description, such that head daughters were lower in the partial order than their mothers. In this case, if all head features had defeasible values, the HFC is given by inheritance. As this operation is order independent it can be interpreted either in terms of rule pre-compilation or interleaved with the construction of a syntactic description. However, there is a potential problem with the persistency of defeasible specifications in this context: whilst it is quite possible to view the *propagation* of head features in terms of default inheritance, their *application* to (fully-specified) lexical items must be indefeasible. Otherwise, lexical information may override head feature specifications leading to analyses of many ungrammatical sentences. Therefore, to make this approach work we would need to ensure that the head features were only interpreted as defeasible with respect to inheritance and were indefeasible with respect to category matching. Clearly, if we simply do this by applying the *DefFill* operation discussed in §4.2, we return to order dependence, because the result will be sensitive to the point at which we transform the TDFS to a TFS. An alternative would be to regard all values as uniformly defeasible, but only information which should be involved in the HFC would be in a mutual specificity relationship, everything else would be unordered (leading to inconsistency during matching of inappropriate categories).

Developing a satisfactory account of GPSG's HFC in terms of $\overset{<>}{\sqcap}$ is, however, beyond the scope of this paper since it would require significant revisions to the syntactic theory in order to utilise the concepts of typed FSs and specificity in this fashion. Nevertheless, although $\overset{<>}{\sqcap}$ as defined in §3 will not give the precise behaviour of the HFC in GPSG, Bouma's (1992) definition of default unification suffers from taking this single specialised application as the paradigm case. In order to capture the intent of 'add conservatively' (Shieber, 1986b) which

$$\begin{bmatrix} \textbf{base} \\ \text{PHON} : \begin{bmatrix} \text{STEM} : \boxed{1} \end{bmatrix} \\ \text{SYNSEM} : \boxed{2} \end{bmatrix} \mapsto \begin{bmatrix} \textbf{third-person-sing} \\ \text{PHON} : \begin{bmatrix} \text{STEM} : \boxed{1} \end{bmatrix} \\ \text{SYNSEM} : \boxed{2} \end{bmatrix}$$

Figure 24: Third Person Singular Lexical Rule

was introduced solely to capture specific effects of the HFC, Bouma defines default unification to 'weaken' reentrancy statements even in the absence of conflicting defaults (see e.g. his (13)). This has the unfortunate consequence that his definition does not reduce to monotonic unification in the absence of conflict (Carpenter, 1993; Copestake, 1993), severely limiting its application to other problems.

6.5 Lexical Rules

The use of AVM boxed 'reentrancy' notation between subparts of FSs in constraint-based formalisms such as HPSG in the statement of lexical rules is quite arbitrary when viewed from the perspective of information flow between signs and semantically incoherent since the two FSs involved cannot be treated as a single DAG. The attraction of utilising default unification rather than reentrancy to specify patterns of inheritance between signs (and their subparts) is that some of this arbitrariness and incoherence in constraint-based terms might be eradicated, because we could state that information is shared *unless* stated to the contrary and that the derived lexical sign is obtained by default unification of information from the basic sign with that stipulated for the derived one.[17]

We will briefly illustrate the point by defining a restricted notion of lexical rule in terms of $\overset{<>}{\sqcap}$. Throughout this paper we have assumed a definition of lexical rule effectively equivalent to that of Pollard and Sag (1987), in which information is 'copied' from base sign to derived sign by stipulation. A more natural and less arbitrary definition might utilise default unification between base and derived signs and stipulate modifications as far as possible through type constraints on the derived sign. Then information flow between base and derived sign will be a consequence of the persistent default unification of the instantiated 'antecedent' TFS with the specification of the 'consequent' TFS. For example, assuming a lexical rule based treatment of English inflectional morphology, we could represent the third person singular rule as in Figure 24. The intended interpretation of the rule, following Pollard and Sag (1987) is that signs of type **base** which unify with the antecedent TFS in the rule are used to create derived signs of type **third-person-sing**. The 'reentrancies' indexed one and two pass over information from the instantiated antecedent TFS to the consequent. The value of SYNSEM given by the type **base** is assumed to be underspecified with respect to agreement, but the type **third-person-sing** will specify agreement and suffixation.

The interpretation of the rule violates the spirit of a constraint-based formalism as there is no constraint-based connection between the antecedent and consequent TFSs. Thus, the reentrancies between the two TFSs do not have the same interpretation as those which occur within a single TFS but rather function to copy information between distinct nodes of different TFSs. Linguistically, the rule is inelegant in that what remains the same between base and derived entry must be stipulated in addition to what changes. As we will now

[17]Note that Pollard and Sag (1994) omit shared information from their descriptions of lexical rules for notational convenience.

describe, the third-person-sing rule could be simplified to a simple statement of the form **base** $\mapsto$ **third-person-sing**, by reinterpreting $\mapsto$ in terms of $\stackrel{<>}{\sqcap}$.

The operation of lexical rules involves a matching stage and creation stage. Under the new formulation matching is trivial since the rule specifies a type and any sign of this type or its subtypes will match (unify with) the rule 'antecedent'. The creation of a new sign could be formalised as:

$$DefFill(DefFilter(TDFS1, TDFS2) \stackrel{<>}{\sqcap} TDFS2)$$

where TDFS1 is the matching 'input' TDFS and TDFS2 is the 'output' TDFS and *DefFilter* is an operation which creates a TDFS of a type compatible with the type of TDFS2 (and with features appropriate for that type) in which the information contained in TDFS1 and appropriate for the type of TDFS2 is made default: that is DefFilter outputs a TDFS with no strict information and a defeasible type which is the generalisation of the types of TDFS1 and TDFS2 if those are incompatible. It is necessary to filter the information given in TDFS1 in this way because $\stackrel{<>}{\sqcap}$ would otherwise give a TDFS with the default root node typed $\perp$. DefFilter essentially involves removing all information from TDFS1 which is incompatible with the type of TDFS2 and with its appropriateness conditions: we will not formalise it here, because the details depend on the precise approach to typing taken. Persistent default unification of the new TDFS with TDFS2 ensures that modifications specified in the lexical rule override information from the base sign, but for third-person-sing this would not be needed. *DefFill* ensures that the resultant sign is indefeasible.

The effect of this formulation is that the 'reentrancies' of Figure 24 are replaced by three steps, the first of which removes incompatible information from the base sign, the second of which persistent default unifies the result with the output TDFS specified in the rule, and the third of which makes the result indefeasible. The rule can thus be rewritten simply as:

base $\mapsto$ **third-person-sg**

A further slight complication is that in some cases we want to ignore some parts of the base sign that would survive DefFilter. We can achieve this by making use of Kaplan and Wedekind's (1993) restriction operator. This formalisation considerably simplifies the stipulation of most lexical rules. For example, the dative lexical rule given in Figure 18 above might be reformulated as follows:

transfer-ditrans-ppto-verb $\setminus$ SYN $\mapsto$ **recip-dative-verb**

Here $\setminus$ SYN indicates that the feature SYN is ignored.

The account of lexical rules we have outlined is sufficient to handle the rules proposed in this paper and in Copestake and Briscoe (1994). Because the components of lexical rules are constraints on FSs, it is possible to use (default) inheritance to capture similarities and redundancies between families of such rules (e.g. Copestake and Briscoe, 1992,1994).

The combination of the restriction, DefFilter and default unification operators is not expressive enough to allow a direct implementation of lexical rules which manipulate list-values of features, such as SUBCAT in HPSG, in complex ways. In view of Carpenter's (1991) proof that such rules considerably increase the generative power of otherwise constrained grammatical formalisms, this may not be a bad result. However, it remains to be seen whether this treatment of lexical rules can be used to characterise passive and other valency-changing or diathesis alternation rules in a linguistically motivated fashion. The approach to the dative construction which we describe in §6.3 together with the reformulation of the relevant lexical rule in this section illustrates how, in principle, all such rules might be reanalysed and reformulated. Detailed linguistic arguments that this general approach would yield a more adequate account than those offered in mainstream lexicalist theories can be found in Pinker (1989) and Goldberg (1992) amongst others.

7 Conclusion

We have defined an order independent version of default unification on typed feature structures. The operation has several desirable properties: it behaves like monotonic unification in the cases where there are no conflicts among the TDFSs; it never fails when the indefeasible information is consistent; and it returns a single, deterministic result. It extends that of Young and Rounds (1993) in two important respects. First, it handles default reentrancies: unifying paths with distinct values with a default reentrancy doesn't lead to unification failure. Second, it enables defaults on TDFSs with more specific types to override conflicting defaults on more general types.

We extended the definition of typed feature structures given in Carpenter (1992), so that a TDFS carries with it, some information from its unification history. We showed that this information is essential for order independence with the full language, but we also described sublanguages for which less information need be retained. The semantics of the persistent default unification operation are defined in a modal conditional logic. This allows us to formally link lexical defaults with pragmatic processing. We left as future work some formal issues concerning the relationship of type constraints and defaults: this will be undertaken in the context of the particular approach to typing and constraints used in the ACQUILEX LKB system (Copestake *et al.*, 1993). We should also note that although the nature of the problem of ensuring order independence determined the basic framework of our definition of persistent default unification (in particular the retention of the unification history in the form of tails) there were some subsidiary decisions where alternative options could be considered. In particular, alternative treatments of default reentrancy seem possible, although there would be a cost in terms of conceptual complexity and computational tractability, if we moved to a more expressive definition where reentrancy could be overridden by default values. As with the definition of the sublanguages, the formal framework we have developed makes it relatively straightforward to investigate variants on the main definition presented here, if required for particular applications.

We demonstrated the utility of persistent default unification in several linguistic applications: inheritance in the lexicon, feature propagation in syntax, semantic broadening of lexical items, the dative construction and the formulation of lexical rules. Moreover, we argued that because default information persists as default under the operation, defaults can survive the lexicon and potentially be overridden by compositional semantic rules or by pragmatic information at the discourse level.

Acknowledgements

This work has been presented at the Centre for Cognitive Science, University of Edinburgh; Rank Xerox Research Centre, Grenoble; University of Pittsburgh; CSLI, Stanford University;

and the Workshop on Universals in the Lexicon in Dagstuhl, Germany. We are grateful for the comments from the participants of these seminars. In addition, we would like to thank Chris Brew, Bob Carpenter, Gabriella Crocco, Tomaz Erjavec, Roger Evans, Gerald Gazdar, Olivier Gasquet, Ron Kaplan, Lauri Kartunnen and Carl Vogel and one anonymous reviewer for helpful comments and advice. This work was partly supported by the ESPRIT Acquilex-II, project BR-7315, grant to Cambridge University. Xerox PARC very kindly provided Ann Copestake with working facilities.

References

Asher, N. and A. Lascarides (1994, in press) 'Lexical Disambiguation in a Discourse Context', *Journal of Semantics,*

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. .

Bird, S. and E. Klein (1994) 'Phonological Analysis in Typed Feature Systems', *Computational Linguistics, vol.20.3,*

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. (1990) 'Defaults in unification grammar', *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics (ACL-90),* Pittsburg, Pennsylvania, pp. 165–173.

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

Boutilier, C. (1992) *Conditional Logics for Default Reasoning and Belief Revision,* PhD Thesis, The University of British Columbia, Technical Report 92-1.

Brewka, G. (1991) 'Cumulative Default Logic: In Defense of Nonmonotonic Inference Rules', *Artificial Intelligence, vol.50,* .

Briscoe, E.J. (1993) 'Introduction' in E. J. Briscoe, A. Copestake and V. de Paiva (eds.), *Inheritance, Defaults and the Lexicon,* Cambridge University Press, Cambridge, England, pp. 1–12.

Briscoe, E.J., C. Grover, B.K. Boguraev and J. Carroll (1987) 'A Formalism and Environment for the Development of a Large Grammar of English', *Proceedings of the 10th Int. Joint Conf. on Artificial Intelligence (IJCAI-87),* Milan, Italy, pp. 703–708.

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.

Briscoe, E.J., A. Copestake and A. Lascarides (1993, in press) 'Blocking' in P. St. Dizier, P. and E. Viegas (eds.), *Computational Lexical Semantics,* Cambridge University Press, Cambridge, England.

Calder, J. (1989) 'Paradigmatic morphology', *Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics (EACL-89),* Manchester, England, pp. 58–65.

Carpenter, R. (1991) 'The Generative Power of Categorial Grammars and Head-Driven Phrase Structure Grammars with lexical Rules', *Computational Linguistics, vol.17.3,* 301–314.

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 ACQUILEX LKB: representation issues in semi-automatic acquisition of large lexicons', *Proceedings of the 3rd Conference on Applied Natural Language Processing (ANLP-92),* Trento, Italy, pp. 88–96.

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.

Copestake, A. and E.J. Briscoe (1992) 'Lexical Operations in a Unification Based Framework' in J. Pustejovsky and S. Bergler (eds.), *Lexical Semantics and Knowledge Representation; Proceedings of the 1st SIGLEX Workshop, Berkeley, California,* Springer Verlag, Berlin, pp. 101–119.

Copestake, A. and E.J. Briscoe (1994, in press) 'Semi-Productive Polysemy and Sense Extension', *Journal of Semantics,*

Copestake, A., A. Sanfilippo, E.J. Briscoe and V. de Paiva (1993) 'The ACQUILEX LKB: an Introduction' in E. J. Briscoe, A. Copestake and V. de Paiva (eds.), *Inheritance, Defaults and the Lexicon,* Cambridge University Press, Cambridge, England, pp. 148–163.

Cruse, D.A. (1986) *Lexical Semantics,* Cambridge University Press, Cambridge, England.

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.

Dalrymple, M., J. Lamping and V. Saraswat (1993) 'LFG semantics via constraints', *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics (EACL-93),* Utrecht, The Netherlands, pp. 97–105.

Delgrande, J. P. (1988) 'An Approach to Default Reasoning Based on First Order Conditional Logic: Revised Report', *Artificial Intelligence, vol.36, 1,* 63–90.

Dowty, D. (1989) 'On the semantic content of the notion 'Thematic Role'.' in G. Chierchia, B. Partee and R. Turner (eds.), *Property Theory, Type Theory and Natural Language Semantics,* D. Reidel, Dordrecht, The Netherlands, pp. 69–129.

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.

Fillmore, C.,P. Kay and C. O'Connor (1988) 'Regularity and idiomaticity in grammatical constructions', *Language, vol.64,* 501–538.

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.

Gazdar, G. (1987) 'Linguistic applications of default inheritance mechanisms' in P. Whitelock, H. Somers, P. Bennett, R. Johnson, and M. McGee Wood (eds.), *Linguistic Theory and Computer Applications,* Academic Press, London, pp. 37–68.

Gazdar, G., E. Klein, G. Pullum and I. Sag (1985) *Generalized Phrase Structure Grammar,* Basil Blackwell, Oxford.

Gerdemann, D. and P. King (1994) 'The Correct and Efficient Implementation of Appropriateness Specifications for Typed Feature Structures', *Proceedings of the 15th Int. Conf. on Comp. Ling. (COLING-94),* Kyoto, Japan.

Goldberg, A. (1992) *Argument structure constructions,* PhD Thesis, UC Berkeley.

Grover, C., Brew, C., Manandhar, S. and Moens, M. (1994) 'Priority Union and Generalization in Discourse Grammars', *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics,* Las Cruces, pp. 17–24.

Grover, C., Brew, C., Manandhar, S. and Moens, M. (1994) 'Priority Union and Generalization in Discourse Grammars', *Proceedings of the Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics,* Las Cruces, New Mexico, pp. 35–41.

Halvorsen, P.K. and R. Kaplan (1988) 'Projections and semantic description in LFG', *Proceedings of the Int. Conf. on 5th Generation Computer Systems,* Tokyo, pp. 1116–1122.

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.

Kaplan, R. and J. Wedekind (1993) 'Restriction and Correspondence-Based Translation', *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics (EACL-93),* Utrecht, The Netherlands, pp. 193–202.

Karttunen, L. (1986) 'D-PATR: a development environment for unification-based grammars', *Proceedings of the 11th Int. Conf. on Comp. Ling. (COLING-86),* Bonn, Germany, pp. 74–80.

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.

Konolige, K. (1988) *Hierarchic Autoepistemic Theories for Nonmonotonic Reasoning: Preliminary Report,* Technical Note No. 446, SRI International.

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. (1995) *The Pragmatics of Word Meaning, Proceedings of the AAAI Spring Symposium on Lexical Semantics*, Stanford CA.

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

Lascarides, A. and Copestake, A. (1995) *A Logic for Order Independent Typed Default Unification,* ACQUILEX research report, also submitted to ACL95.

Maxwell, J.T. and R. Kaplan (1993) 'The interface of functional and syntactic constraints', *Computational Linguistics, vol.19.4,* 571–590.

Morreau, M. (1992) *Conditionals in Philosophy and Artificial Intelligence,* PhD Thesis, IMS, Universität Stuttgart.

de Paiva, V. (1993) 'Types and Constraints in the LKB' in E. J. Briscoe, A. Copestake and V. de Paiva (eds.), *Inheritance, Defaults and the Lexicon,* Cambridge University Press, Cambridge, England, pp. 164–189.

Parsons, T. (1990) *Events in the Semantics of English: a Study in Subatomic Semantics,* MIT Press, Cambridge, Massachusetts.

Pinker, S (1989) *Learnability and Cognition: The Acquisition of Argument Structure,* MIT Press, Cambridge, MA.

Pollard, C. and I. Sag (1987) *An information-based approach to syntax and semantics: Volume 1 fundamentals,* CSLI Lecture Notes 13, Stanford CA.

Pollard, C. and I. Sag (1994) *??,* CSLI Lecture Notes, Stanford, CA.

Pustejovsky, J. (1991) 'The Generative Lexicon', *Computational Linguistics, vol.17.4,* 409–441.

Pustejovsky, J. and B. Boguraev (1993) 'Lexical knowledge representation and natural language processing', *Artificial Intelligence, vol.63,* 193–223.

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.

Russell, G., A. Ballim, J. Carroll and S. Warwick-Armstrong (1993) 'A Practical Approach to Multiple Default Inheritance for Unification-based Lexicons' in E. J. Briscoe, A. Copestake and V. de Paiva (eds.), *Inheritance, Defaults and the Lexicon,* Cambridge University Press, Cambridge, England, pp. 137–147.

Sanfilippo, A. (1990) *Grammatical Relations, Thematic Roles and Verb Semantics,* PhD thesis. Centre for Cognitive Science, University of Edinburgh.

Sanfilippo, A. (1993) 'LKB encoding of lexical knowledge' in E. J. Briscoe, A. Copestake and V. de Paiva (eds.), *Inheritance, Defaults and the Lexicon,* Cambridge University Press, Cambridge, England, pp. 190–222.

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

Shieber, S. M. (1986b) 'A simple reconstruction of GPSG', *Proceedings of the 11th Int. Conf. on Comp. Ling. (COLING-86),* Bonn, Germany, pp. 211–215.

Smolka, G. and H. Aït-Kaci (1988) 'Inheritance hierarchies: Semantics and unification', *Journal of Symbolic Computation, vol.7,* 343–370.

Sweetser, E. (1990) *From etymology to pragmatics,* Cambridge University Press, Cambridge, England.

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.

Zajac, R. (1993) 'Issues in the design of a language for representing linguistic information based on inheritance and feature structures' in E. J. Briscoe, A. Copestake and V. de Paiva (eds.), *Inheritance, Defaults and the Lexicon,* Cambridge University Press, Cambridge, England, pp. 74–89.