A Model and Representation for Type Information and Its Use in Reasoning with Defaults

by

Lin Padgham

RESEARCH REPORT
RKLLAB, June 1988

Postaddress:
Institutionen för datavetenskap
Universitetet i Linköping och
Tekniska Högskolan
581 83 Linköping

Mailing address:
Department of Computer and
Information Science
Linköping University
S-581 83 Linköping, Sweden
The Department of Computer and Information Science
Linköping University

PhD theses:

* Available at: University of Microfilms Intl., 300 N. Zeeb Road, Ann Arbor, MI 48106, USA.

( Linköping Studies in Science and Technology. Dissertations. )

No 97 Andrzej Lingas: Advances in Minimum Weight Triangulation, 1983.
*No 109 Peter Fritzson: Towards a Distributed Programming Environment based on Incremental Compilation, 1984.

Licentiate of engineering Theses:

( Linköping Studies in Science and Technology. Theses. )

No 73 Ola Strömfors: A Structure Editor for Documents and Programs. 1986.
No 108 Rober Bilos: Incremental Scanning and Token-based Editing. 1987
No 113 Ralph Rönnquist: Network and Lattice Based Approaches to the Representation of Knowledge. 1987.
No 118 Mariam Kamkar, Nahid Shahrmehr: Affect-Chaining in Program Flow Analysis Applied to Queries of Programs. 1987
No 126 Dan Strömberg: Transfer and Distribution of Application Programs. 1987
A Model and Representation for Type Information and Its Use in Reasoning with Defaults

by

Lin Padgham

Abstract: Typing schemes which allow inheritance from super- to sub-types are a common way of representing information about the world. There are various systems and theories which use such representations plus some inferencing rules to deduce properties of objects, about which the system has only partial information. Many of these systems have problems related to multiple inheritance, and have some difficulty in drawing conclusions which we as humans see as intuitively simple.

We present a model of typing based on a lattice of feature descriptors. A type is represented by two important points in the lattice representing core and default information. The use of two points allows some information to be monotonic whilst other information is nonmonotonic.

We give some operations which can be used in default reasoning about an object, based on knowledge about the relationships between the points in the lattice which are defined as types. We then work through some specific examples, showing the conclusions which we reach with this system. We compare the expressiveness of our system to some of the well known work in the area of default reasoning using inheritance.

Postaddress:
Institutionen för datavetenskap
Universitetet i Linköping och
Tekniska Högskolan
581 83 Linköping

Mailing address:
Department of Computer and
Information Science
Linköping University
S-581 83 Linköping, Sweden
1. Introduction

Classification into categories and sub-categories, along with default reasoning about the properties of objects within these categories is a common human activity. It allows for continuation of the reasoning process in situations where people could otherwise be paralyzed by lack of information, or by the overwhelming number of theoretical possibilities.

There have been a number of systems built which attempt to capture the notion of inheritance from super to sub-types, and to simulate the human reasoning regarding characteristics of objects known to belong to a certain type. Some well known examples are FRL [ROB77], NETL [FAH79] and TMOIS [TOU86]. These systems all have difficulties around issues to do with multiple inheritance combined with exceptions. They easily run into ambiguous situations where they cannot make a decision, or make an intuitively wrong decision. This is true even for questions which people would resolve easily and unambiguously.

We present a model of typing based on the splitting of a type into type default and type core, and on the formalism of a lattice, rather than the more usual notions of a tree or an acyclic directed graph ("tangled hierarchies" [FAH 79]). Our model allows for representation of information which we believe people typically use in reasoning about defaults, and which is not representable in the above mentioned systems. This model plus the inference mechanism we suggest gives clear solutions to many of the problems which have previously been difficult. It also provides solutions in a consistent manner, without resorting to more complex inference mechanisms to deal with special situations.

2. Cores and Defaults

We use the notion of a type being defined not as a node in a graph of super and subtypes, but rather as a collection of characteristics which we expect to see in an object of a particular type. It is our view that this is a natural description of what typing or classification is for humans. It is the association of certain characteristics together as a group.

If we then look at the characteristics defining a type, we see that we are far more willing to override some of the characteristics than others. For this reason we identify two clusterings representing a type, — the type core and the type default.

The type core includes those characteristics which we regard as always present in objects of this type. 1 Only type descriptions which are above2 this type core are considered to be subtypes of this type. If we state that A is a subtype of B then we know unequivocally that the core descriptor for type A contains at least all the information that is contained in the core descriptor for B.

1 Individual objects can always fall below even the type core. This is taken up further in an example. We are currently working on a suitable way of describing such objects in order to reason about them in the way intuitively desired.

2 Note that specialization or more information is higher in the lattice, whereas it is lower in inheritance graphs.
The *type default* contains the information for typical objects of that type. The type default must of course contain at least the same information as is contained in the type core.

An individual object can also be seen as a clustering of characteristics which we can call the *descriptor* for that object.

Using this notion of two points to describe a type, let us look at a simple but classical problematic example.

### 2.1 Example 1

![Diagram](image)

**Traditional Representation**

The network 1a with its redundant link (Clyde → Elephant) is given by Touretsky [TOU87] as the sort of example that causes shortest path reasoning algorithms to fail, and is thus a justification for his shortest inferential distance (or on-path preemption) algorithm. Sandewall [SAN86] shows with network 1b that shortest inferential distance is also inadequate and proposes an algorithm which Touretsky [TOU87] refers to as offpath preemption. Our approach gives the desired result in both examples, and is appealing in that the addition of extra information (links to and from African Elephant) does not require any change in reasoning strategy.

We can write the following relationships from our representation:

\[
\begin{align*}
\text{Core}(\text{Royal Elephant}) & \supseteq \text{Core}(\text{Elephant}) \\
\text{Core}(\text{African Elephant}) & \supseteq \text{Core}(\text{Elephant}) \\
\text{Default}(\text{Elephant}) & \supseteq \text{Core}(\text{Grey thing})
\end{align*}
\]

\[
\begin{align*}
\text{Core}(\text{Royal Elephant}) \cup \text{Core}(\text{Grey thing}) & \Rightarrow K \\
\text{Core}(\text{Royal Elephant}) & \in \text{NOT}(\text{Core}(\text{Grey thing})) \\
\text{Core}(\text{Grey thing}) & \in \text{NOT}(\text{Core}(\text{Royal Elephant}))
\end{align*}
\]

2 Note that we have represented the information that royal elephants must be non-grey, whereas only typical elephants are grey.
\[ \text{Desc(Clyde)} \sqsupseteq \text{Core(African Elephant)} \]
\[ \text{Desc(Clyde)} \sqsupseteq \text{Core(Royal Elephant)} \]

Our diagram defines a set of **inheritance paths** having a specific object (e.g. Clyde) as their first element, and which characterize properties the object may inherit. The inheritance paths are written in the form:

\[ n_0 \, op \, n_1 \, op \, n_2 \, op \, \ldots \, n_i \, op \, n_{i+1} \, \ldots \]

where the operation \( op \) in the link \( n_i \, op \, n_{i+1} \), may be either of the following:

\[ \supseteq \]
\( n_i \supseteq n_{i+1} \) may occur in the path if the corresponding relationship occurs in the diagram.

\[ \in \]
\( \text{Core}(A) \in \text{Default}(A) \) may occur in the path, representing a default assumption that the object at the beginning of the path is not only an \( A \), but is a typical \( A \), (in so far as \( A \)'s typicality does not conflict with already known information).

Notice that path steps using \( \supseteq \) proceed downwards in the graph, and steps using \( \in \) proceed upwards, thus adding extra information to \( n_0 \) besides what is deductively available.

\[ \subset \]
The path may be terminated by \( n_i \in \text{NOT}(A) \), indicating that \( n_i \) is a member of the set of nodes:

\[ \text{NOT}(A) \equiv \{ x \mid x \cup A \Rightarrow K \} \quad (K \equiv \text{inconsistent in at least one feature value}) \]

A set of paths is said to be inconsistent if it contains both the step \( \supseteq A \), and the step \( \in \text{NOT}(A) \), and consistent otherwise. The full set of paths obtained by using the \( \in \) operator wherever possible is often inconsistent. Consequently we only allow adding in of information following an \( \in \) operation provided it is not inconsistent with information obtained at an earlier step.

We can now infer the following about Clyde:

\[ \text{Desc(Clyde)} \supseteq \text{Core(African Elephant)} \supseteq \text{Core(Elephant)} \]
\[ \ldots \text{Core(African Elephant)} \supseteq \text{Core(Elephant)} \in \text{Default(Elephant)} \]
\[ [ \supseteq \text{Core(Grey thing)} ] \]

\[ \text{Desc(Clyde)} \supseteq \text{Core(Royal Elephant)} \supseteq \text{Core(Elephant)} \]
\[ \ldots \text{Core(Royal Elephant)} \in \text{NOT(Core(Grey thing))} \]

We see here that we have a single extension, with no default assumptions, giving the result that Clyde, in addition to being a Royal Elephant and an African Elephant, is an Elephant and NOT a Grey thing.

If we had instead represented and reasoned about the information for the network 1a, we would simply have been missing the information:

\[ 2 \text{ We show in } [ ] \text{ the information which potentially could have been added in following the extension operation, but which is not added due to its being contradictory.} \]
\[ \text{Desc(Clyde)} \equiv \text{Core(African Elephant)} \]
\[ \text{Core(African Elephant)} \equiv \text{Core(Elephant)} \]

This would not have altered the inferred information. The missing initial information that Clyde is an African Elephant would of course also be missing in the conclusion.

3. A Lattice Based Model of Types

Taking the theoretical space of all possible combinations of all possible characteristics induces a lattice of descriptors. Any clustering of characteristics then belongs somewhere in this lattice. Typing can be seen as naming some of the points in this lattice as being relevant clusterings of characteristics.

This view allows for both descriptive and prescriptive typing. We can deduce the type(s) of an object by having information concerning its characteristics, and we can deduce information about an object's characteristics by having information about its types. Human reasoning also uses typing and classification in both these ways.

Reasoning about the relation between individual objects and type descriptors can also be applied to relations between subtypes and their supertypes. A particular type, A, can be observed to be a specialization or subtype of some other type, B, if the type descriptor for A contains all the information in the type descriptor for B plus some extra information.

Conclusions can be drawn about the type(s) of a given object by placing its descriptor in the lattice, and noting which types it falls above.

However we do not always have complete information regarding an object, and can therefore not place its descriptor directly in the lattice. We introduce the notion of partial information in the descriptors, plus relations between the descriptors which constrain their positioning in the lattice without fully defining it. If we say the descriptor for A is above the descriptor for B (\( A \equiv B \)), and we know that P is an A, then we can conclude (by transitivity of \( \equiv \)) that P is also a B, without knowing any of the actual characteristics of either descriptor.

By comparing two types with respect to cores and defaults, we can make the statement A's are B's with four different shades of meaning, ordered with respect to the strength of the statement.

\[ \text{Core}(A) \equiv \text{Default}(B) \]
A's are always typical B's.

\[ \text{Core}(A) \equiv \text{Core}(B) \]
A's are always B's – but not necessarily typical B's.

\[ \text{Default}(A) \equiv \text{Default}(B) \]
A's are usually typical B's.
Default(A) \equiv \text{Core}(B)

A's are usually B's - but not necessarily typical B's.

The greater expressivity given by split into core and default is often valuable in reasoning about characteristics of objects. The information is intuitive for humans (at least within some fuzzy boundaries), and needs to be represented if a reasoning system is to draw conclusions we intuitively wish it to draw. This is demonstrated in examples.

4. Representation of negative information

Multiple inheritance reasoning systems often have negative links in the inheritance graph, which are important in the reasoning process. (E.g. NETL [FAH79]and TMOIS[TOU86]. On examination, these negative links are used to express two different things. These can be described as incompatibility and overriding. We define incompatibility within the lattice framework, and show overriding to be unnecessary.

4.1 Incompatibility

By incompatibility between A and B we mean that there is at least one characteristic of A that is incompatible with a characteristic of B. Consequently no object can be both an A and a B. This can be described as

\[ A \sqcup B \Rightarrow \kappa \]

where \( A \sqcup B \) is the least upper bound (or join) of A and B, and \( \kappa \) indicates that at least one feature value is inconsistent. Incompatibility can also be expressed as \( A \in \text{NOT}(B) \).

A specialization of incompatibility is the case where the descriptor consists essentially of a single property which has the value TRUE for A and FALSE for B. For example 'Shellbearing' and 'NOT Shellbearing' can be seen as two descriptors one with the property Shell, value TRUE, the other with the property Shell, value FALSE.

As with positive relations between types we can use the core and default for the type in order to say "A's are not B's" with different shades of meaning.

\text{Core}(A) \sqcup \text{Core}(B) \Rightarrow \kappa

A's are never B's.

\text{Core}(A) \sqcup \text{Default}(B) \Rightarrow \kappa

A's are never typical B's.

\text{Default}(A) \sqcup \text{Core}(B) \Rightarrow \kappa

Typical A's are not B's.

\text{Default}(A) \sqcup \text{Default}(B) \Rightarrow \kappa

Typical A's are not typical B's.
4.2 Overriding

Overriding refers to the situation where in NETL[FAH79] and TMOIS[TOU86] negative links are used to override positive inherited information. The situation is not such that the negatively linked points are necessarily incompatible, but rather that it is necessary to block a possible chain of reasoning.

Within our framework this situation occurs when there exists some known type, B, which is above Core(A) but not above Default(A), and B is incompatible with Default(A). One then wants to ensure that the chain of reasoning from Default(A) is disallowed. Because we make a distinction between core and default and can identify subtypes that are known to lie between the core and default, we do not need any special override mechanism.

5. Reasoning Using The Lattice Information

Having defined how to express information regarding types and their relations to each other, we use the inference mechanism as described in example one, for reasoning with the information to state what types a given object may have.

The basic relation \( \sqsupseteq \) between lattice points is transitive, so if we know that an object P has a descriptor such that \( \text{Desc}(P) \sqsupseteq \text{Core}(A) \), and that \( \text{Core}(A) \sqsupseteq \text{Core}(B) \), we can conclude that \( \text{Desc}(P) \sqsupseteq \text{Core}(B) \). This method allows us to state all definite (monotonic) positive conclusions regarding \( \text{Desc}(P) \). We can of course always add relations of the form \( \text{Default}(x) \sqsupseteq \text{Core}(x) \) (by definition).

Negative information of the form \( 'A \sqsubseteq B \Rightarrow K' \), gives us relations of the form \( 'X \sqsupseteq A \Rightarrow X \in \text{NOT}(B)' \) as defined previously.

The extension operation \( \varepsilon \) defined in example one allows us to make the nonmonotonic jump from the core of a given type to its default,

\[
\text{Core}(A) \varepsilon \text{Default}(A)
\]

The extension operation can be repeated as long as it is possible to obtain more information by doing so. Each such operation implies an assumption. If there is more than one such assumption in the reasoning process, the ordering of the extension operation gives the different valid extensions. Preference for making this assumption at the most specific point possible (in terms of the \( \sqsupseteq \) lattice relation) gives the intuitively desirable preferences between extensions.

This preference for the most specific Core to Default specializations prefers the same extensions as those obtained by the shortest inferential distance algorithm of Touretsky, and the offpath preemption algorithm of Sandewall, but in a more clearly motivated and consistent manner.
6. Some Classical Examples

We will now work through some further examples from the literature in order to demonstrate that our lattice based approach with separate core and default produces the desired results.

6.1 Example 2

Here we see the same network as in example 1b, (given previously) but with different labeling on the nodes. Touretsky argues that with the changed labeling it is less intuitively clear what conclusions we wish to draw. The lattice diagram shows the suggested representation within our model, which differs from the lattice representation given for example 1. This captures the differing strength in the information that Marines (and Chaplains) are Men, compared with African Elephants (and Royal Elephants) are Elephants.

Extracting the formulas similarly to example 1, and abbreviating Core to C and Default to D, we reason as follows:

George ⊇ C(Marine) ⊆ D(Marine) ⊇ C(Man) ⊆ D(Man) ⊇ C(Beerdrinker)
George ⊇ C(Chaplain) ⊆ D(Chaplain) ⊇ C(Man)
...D(Chaplain) ⊆ NOT(C(Beerdrinker))

The extension to Default(Man), giving Core(Beerdrinker) conflicts with the extension to Default(Chaplain) which gives ∈ NOT(Beerdrinker). Ordering of the conflicting extension operations gives two extensions — one in which George is a Beerdrinker and one in which he is NOT a Beerdrinker. The previously discussed preference for extension operations from the highest lattice points, when there is a conflict, gives preference to the extension in which George is NOT a Beerdrinker. (Core(Chaplain) ⊇ Core(Man))

The representation that allows us to differentiate between 'Typical Chaplains are Men' vs 'All Royal Elephants are Elephants' enables us to then use this information to
reason in a more natural (and correct?) way, than the IS—A network representation does.

6.2 Example 3

This example is the canonical ambiguous net ('the Nixon diamond') concatenated with a net giving further ambiguities (from TOU87). We work through this example with two different choices of representation in our model, in order to bring out some properties of our model.

6.2.1 Quakers are always Pacifists (3b)

In the following representation we represent all Quakers as being Pacifists, but only default Republicans as being incompatible with Pacifists. All other choices between default and core of the types have no influence on the resulting conclusions. (They simply influence the certainty of the extensions. We have made choices here to reduce the number of extension operations needed)

By using the formulas as previously we reason as follows:

\[ \text{Nixon} \supseteq C(\text{Quaker}) \supseteq D(\text{Pacifist}) \supseteq C(\text{Pacifist}) \supseteq C(\text{Antimilitary}) \]
\[ \text{Nixon} \supseteq C(\text{Repub.}) \not\subset D(\text{Repub.}) \supseteq D(\text{Football fan}) \ [ \in \text{NOT}(C(\text{Antimilitary})) ] \]
\[ ...D(\text{Repub.}) \ [ \in \text{NOT}(C(\text{Pacifist})) ] \]

We obtain a single extension in which Nixon is a Quaker, Republican, Pacifist, Antimilitary and Football fan. This is disturbing to our intuitions as Nixon is a particular individual who we know to have been non—pacifist and non—antimilitaristic. This serves to illustrate the point previously noted, that individuals must be allowed to fall even below the type core, where explicitly stated to do so. However this does not justify the building of a hypothetical class of militaristic Quakers. We are currently working on an appropriate representation for individuals who fall below the core, that will enable us to continue the desired reasoning about such an individual.
6.2.2 Typical Quakers are Pacifists (3c)

In this representation we replace the original symmetry of the Nixon diamond by stating that only typical Quakers are Pacifists. We discuss the conclusions drawn with reference to Touretsky's discussion of cascading ambiguities, and skeptical vs. credulous reasoners.

Doing the Quaker Core to Default $\xi$ first:

\[
\begin{align*}
\text{Nixon} & \sqsupseteq C(\text{Quak.}) \thicksim D(\text{Quak.}) \sqsupseteq D(\text{Pac.}) \sqsupseteq C(\text{Pac.}) \sqsupseteq C(\text{Antimil.}) \\
\text{Nixon} & \sqsupseteq C(\text{Rep.}) \thicksim D(\text{Rep.}) \sqsupseteq D(\text{Foot. fan}) \in \text{NOT}(C(\text{Antimil.}))) \\
& \quad \text{...}(\text{Rep.}) \in \text{NOT}(C(\text{Pacifist}))
\end{align*}
\]

or, doing the Republican Core to Default $\xi$ first:

\[
\begin{align*}
\text{Nixon} & \sqsupseteq C(\text{Quaker}) \thicksim D(\text{Quaker}) \sqsupseteq \{ D(\text{Pacifist}) \} \\
\text{Nixon} & \sqsupseteq C(\text{Rep.}) \thicksim D(\text{Rep.}) \sqsupseteq D(\text{Foot. fan}) \in \text{NOT}(C(\text{Antimil.})) \\
& \quad \text{...}(\text{Rep.}) \in \text{NOT}(C(\text{Pacifist}))
\end{align*}
\]

We obtain two extensions, one the same as with the previous representation, and one in which Nixon is Quaker, Republican, Football fan, NOT Pacifist and NOT Antimilitary.

If we had represented the link from Pacifist to Antimilitary as weaker than we did (i.e. as Default(Pacifist) $\sqsupseteq$ Core(Antimilitary)) we would also have generated an extension in which Nixon was Quaker, Republican, Football fan, Pacifist and NOT Antimilitary.

The results here are consistent with Touretsky's description of a credulous reasoner, in that all possible extensions are found. In this case there is no simple preference between extensions based on assumption of typicality at the most specific point possible. However all extensions are obtained if one wishes to then choose among extensions on the basis of various heuristics.

6.3 Example 4

\[
\begin{align*}
\text{NETL} & \quad \text{Etheringtons} & \quad \text{Lattice}
\end{align*}
\]

\[
\begin{align*}
\text{Representation} & \quad \text{Representation} & \quad \text{Representation}
\end{align*}
\]
Here we show an example from Fahlman et al (FAH81). We show it both represented as it is in NETL, and also with the more informative representation of Etherington, who uses a wider variety of link types. Etherington's link types have the following meanings.

\[
\begin{array}{c}
\text{Strict} \\
\text{IS-A} \\
\end{array} \quad \begin{array}{c}
\text{Strict} \\
\text{IS-NOT-A} \\
\end{array} \quad \begin{array}{c}
\text{Default} \\
\text{IS-A} \\
\end{array} \quad \begin{array}{c}
\text{Default} \\
\text{IS-NOT-A} \\
\end{array} \quad \text{Exception}
\]

The lattice representation comes directly from Etherington's network and does not contain any extra information that is intuitively assumed in choosing between core and default. The information is already implicit in the graph.

Reasoning about the individual Nautilus, X, we get:

\[
X \models C(\text{Nautilus}) \sqsupseteq C(\text{Cephalopod}) \sqsupseteq C(\text{Mollusc})
\]

\[
\ldots C(\text{Cephalopod}) \sqsubseteq D(\text{Cephalopod}) \left[ \in \text{NOT}(C(\text{Shellbearer})) \right]
\]

\[
\ldots C(\text{Mollusc}) \sqsubseteq D(\text{Mollusc}) \sqsupseteq C(\text{Shellbearer})
\]

\[
\ldots C(\text{Nautilus}) \sqsupseteq C(\text{Shellbearer})
\]

We see that an individual Nautilus is, both without competing extensions and also without uncertain assumptions, a Cephalopod, a Mollusc and a Shellbearer. This is the same conclusion that would be reached using either NETL or Etherington's network representation plus his mapping of this to Reiter's default logic.

Reasoning about the individual Cephalopod, Y, we get:

\[
Y \models C(\text{Cephalopod}) \sqsupseteq C(\text{Mollusc}) \sqsubseteq D(\text{Mollusc}) \sqsupseteq C(\text{Shellbearer})
\]

\[
\ldots C(\text{Cephalopod}) \sqsubseteq D(\text{Cephalopod}) \in \text{NOT}(C(\text{Shellbearer}))
\]

This corresponds to two extensions, one in which X is a shellbearer, and one in which it is not. The extension in which it is not a shellbearer is preferred on the basis of its resulting from an extension operation at a more specific point.

This conclusion is neither the same as NETL nor Etherington's conclusions (which also differ from each other here). Etherington's method provides only one extension, the same as our preferred extension. NETL gives both extensions but does not discriminate at all between them. We feel that our solution, providing both extensions, but with a clear and intuitively reasonable preference, is the most desirable.

7. Discussion and Comparison with Other Work

The representation and reasoning methods proposed here appear to offer clear advantages when compared to systems such as TMOIS and NETL which use network representations with two sorts of links (positive and negative) plus possibly exceptions.
The largest gain results from the division of type descriptors into two parts, giving a similar effect to what Touretakty classifies as heterogeneous, bipolar systems. Heterogeneous refers to the ability to have some information which is certain (monotonic), while other is uncertain (nonmonotonic).

Although the lattice itself does not directly represent negative relations, these can be described with respect to the lattice, and used in reasoning. The lattice formalism also gives greater clarity to the semantics of negative links than has been evident in network representations.

The inference mechanism for reasoning about objects within a lattice based type schema appears to be cleaner and more consistent than the inference mechanisms developed for reasoning about network based representations. Many of the examples which have proved problematic for network based systems, and which have required new inference mechanisms, are simple and clear within the lattice model.

A comparison between the lattice based model and the suggested network representations of Etherington (with the corresponding default logic which he suggests) is less definitive with respect to which is better. By increasing the number of link types he also specifies a heterogeneous system. This has similar advantages to our system in that he represents more information, which is necessary for achieving the desired reasoning.

An essential difference between Etherington's model and ours is that he represents explicit information about exceptions, whereas we simply represent lack of typicality. Both approaches have their advantages. The advantage with our model is that if we know there are exceptions we generate both extensions, with a preference for the most likely. Etherington's method generates only the typical extension, unless one knows that the particular exception causing case is present.

We have plans to implement the described system, within LINCKS (Linköpings INtelligent Communication of Knowledge System). An implementation will give us the ability to experiment with larger and more complex examples, and to determine whether the described model does provide the reasoning and representation facilities to achieve intuitively desirable results.

Acknowledgements
We thank Erik Sandewall and Ralph Rönquist for helpful discussions of the ideas presented here, and Jalal Maleki for careful proof reading and useful comments.

References

ETH87
FAH79

FAH81

SAN86

TOU86

TOU87
A Model and Representation for Type Information
and Its Use in Reasoning with Defaults

Lin Padgham

Abstract: Typing schemes which allow inheritance from super- to sub-types are a common way of representing information about the world. There are various systems and theories which use such representations plus some inferencing rules to deduce properties of objects, about which the system has only partial information. Many of these systems have problems related to multiple inheritance, and have some difficulty in drawing conclusions which we as humans see as intuitively simple.

We present a model of typing based on a lattice of feature descriptors. A type is represented by two important points in the lattice representing core and default information. The use of two points allows some information to be monotonic whilst other information is nonmonotonic.

We give some operations which can be used in default reasoning about an object, based on knowledge about the relationships between the points in the lattice which are defined as types. We then work through some specific examples, showing the conclusions which we reach with this system. We compare the expressiveness of our system to some of the well known work in the area of default reasoning using inheritance.
A Selection of Previous Research Reports.

LiTH-IDA-R-88-06 Christer Bäckström: A Representation of Coordinated Actions Characterized by Interval Valued Conditions.
LiTH-IDA-R-88-05 Christer Bäckström: Keeping and Forcing: How to Represent Cooperating Actions.
LiTH-IDA-R-88-04 Wlodek Drabent, Simin Nadji-Tehrani, Jan Maluszynski: Algorithmic Debugging with Assertions.
LiTH-IDA-R-88-02 Ulf Nilsson: Inferring Restricted AND-Parallelism in Logic Programs using Abstract Interpretation.
LiTH-IDA-R-87-26 Jonas Löwgren: Applying a Rapid Prototyping System to Control Panel Dialogues.
LiTH-IDA-R-87-24 Sven Moen: Drawing Dynamic Trees.
LiTH-IDA-R-87-21 Harold W. Lawson, Jr.: Challenges and Directions in Computers and Education.
LiTH-IDA-R-87-20 Krzysztof Kuchcinski, Zebo Peng: Parallelism Extraction from Sequential Programs for VLSI Applications. This paper is to appear in Microprocessing and Microprogramming, the Euromicro Journal, 1988.
LiTH-IDA-R-87-18 Henrik Nordin: Reuse and Maintenance Techniques in Knowledge-Based Systems.
LiTH-IDA-R-87-17 Tony Larsson: Specification and Verification of VLSI Systems Actional Behaviour This is a close version of a paper presented at the 8th international conference on Computer Hardware Description Languages, CHDL, 87.
LiTH-IDA-R-87-15 Nils Dahlbäck: Kommunikation med datorer i naturligt språk - vad är det och vem behöver det?
LiTH-IDA-R-87-10 Andrzej Lingas: On Parallel Complexity of the Subgraph Isomorphism Problem.
organizes undergraduate and graduate studies in Computer Science, Telecommunication and Computer Systems, and Administrative Data Processing. Research activities have an emphasis on advanced software technology and computer systems design and are organized in a number of research laboratories:

- **ACTLAB - Laboratory for Complexity of Algorithms**, which is concerned with the design and analysis of efficient sequential and parallel algorithms, and complexity theory, especially in the areas of computational geometry, data structures on bounded domains and graph algorithms.

- **AIELAB - Artificial Intelligence Environments Laboratory**, which conducts research on representation and manipulation of knowledge, problem solving techniques and expert systems with mechanical engineering applications.

- **ASLAB - Application Systems Laboratory**, which studies design of advanced support systems for interactive use of computers, including tools for automated construction of applications software.

- **CADLAB - Laboratory for Computer-Aided Design of Electronics**, which concentrates its research activities around tools for integrated development of hardware and software, graphics-based modelling and simulation techniques.

- **LIBLAB - Laboratory for Library and Information Science**, which studies methods for access to documents and the information contained in the documents, concentrating on catalogs and bibliographic representations, and on the human factors of library use.

- **LOGPRO - Laboratory for Logic Programming**, which concentrates its research activities around foundations of logic programming, relations to other programming paradigms and methodology.

- **NLPLAB - Natural Language Processing Laboratory**, which conducts research related to the development and use of natural language interfaces to computer software.

- **PELAB - Programming Environments Laboratory**, which works with design of tools for software development, specific functions in such tools and theoretical aspects of programs under construction.

- **RKL LAB - Laboratory for Representation of Knowledge in Logic**, which covers issues and techniques such as non-monotonic logic, probabilistic logic, modal logic and truth maintenance algorithms and systems.

Research Reports 1988

LiTH-ID-A-88-16  **Lin Padgham**: A Model and Representation for Type Information and Its Use in Reasoning with Defaults. Also in Proc. of AAAI'88, American Association for Artificial Intelligence, 1988.


LiTH-ID-A-88-13  **Erik Tengvall**: Ett kartorierat planeringsystem för autonoma farkoster, en design diskussion.


LiTH-ID-A-88-11  **Mats Wirén**: An Incremental Chart Parser for PATR.

LiTH-ID-A-88-10  **Mats Wirén**: A Control-Strategy-Independent Parser for PATR.