A Description Logic with Transitive and Inverse Roles and Role Hierarchies

Ian Horrocks*
Medical Informatics Group,
University of Manchester

Ulrike Sattler†
LuFG Theoretical Computer Science,
RWTH Aachen

1 Motivation

As widely argued [Horrocks & Gough,1997; Sattler,1996], transitive roles play an important rôle in the adequate representation of aggregated objects: they allow these objects to be described by referring to their parts without specifying a level of decomposition. In [Horrocks & Gough,1997], the Description Logic (DL) $\mathcal{ALCH}_{R^+}$ is presented, which extends $\mathcal{ALC}$ with transitive roles and a role hierarchy. It is argued in [Sattler,1998] that $\mathcal{ALCH}_{R^+}$ is well-suited to the representation of aggregated objects in applications that require various part-whole relations to be distinguished, some of which are transitive. For example, a medical knowledge base could contain the following entries defining two different parts of the brain, namely the gyrus and the cerebellum. In contrast to a gyrus, a cerebellum is an integral organ and, furthermore, a functional component of the brain. Hence the role $\mathtt{is\_component}$ (which is a non-transitive sub-role of $\mathtt{is\_part}$) is used to describe the relation between the brain and the cerebellum:

$$
\begin{array}{ll}
\mathtt{is\_component} \sqsubseteq \mathtt{is\_part} & \\
\mathtt{gyrus} \quad := & \\
\quad (\forall \mathtt{consists}.\mathtt{brain\_mass}) \sqcap (\exists \mathtt{is\_part}.\mathtt{brain}) & \\
\mathtt{cerebellum} \quad := & \\
\qquad \mathtt{organ} \sqcap (\exists \mathtt{is\_component}.\mathtt{brain}) &
\end{array}
$$

However, $\mathcal{ALCH}_{R^+}$ does not allow the simultaneous description of parts by means of the whole to which they belong and of wholes by means of their constituent parts: one or other is possible, but not both. To overcome this limitation, we present the DL $\mathcal{ALCHI}_{R^+}$ which extends $\mathcal{ALCH}_{R^+}$ with inverse (converse) roles, allowing, for example, the use of $\mathtt{has\_part}$ as well as $\mathtt{is\_part}$.[1] Using $\mathcal{ALCHI}_{R^+}$, we can define a tumorous brain as:

$$
\begin{array}{l}
\mathtt{tumorous\_brain} := \\
\quad \mathtt{brain} \sqcap (\mathtt{tumorous} \sqcup (\exists \mathtt{has\_part}.\mathtt{tumorous}))
\end{array}
$$

and then to recognise that $\mathtt{cerebellum} \sqcap \mathtt{tumorous}$ is subsumed by $\exists \mathtt{is\_component}.\mathtt{tumorous\_brain}$.

Furthermore, $\mathcal{ALCHI}_{R^+}$ allows for the internalisation of *general inclusion axioms* [Horrocks & Gough,1997].

It could be argued that, instead of defining yet another DL, one could make use of the results presented in [De Giacomo & Lenzerini,1996] and use $\mathcal{ALC}$ extended with role expressions which include transitive closure and inverse operators. The reason for not proceeding like this is the fact that transitive roles can be implemented more efficiently than the transitive closure of roles (see [Horrocks & Gough,1997]), although they lead to the same complexity class (EXPTIME-hard) when added, together with role hierarchies, to $\mathcal{ALC}$. Furthermore, it is still an open question whether the transitive closure of roles together with inverse roles necessitates the use of the *cut rule* [De Giacomo & Massacci,1998], a rule which leads to an algorithm with very bad behaviour. We will present an algorithm for $\mathcal{ALCHI}_{R^+}$ without such a rule, which, from the experiences made with an implementation of $\mathcal{ALCH}_{R^+}$ [Horrocks & Gough,1997], should behave well in practice.[2]

2 Blocking

The algorithms which we will present use the tableaux method, in which the satisfiability of a concept D is tested by trying to construct a model of D. The model is represented by a tree in which nodes correspond to individuals and edges correspond to roles. Each node x is labelled with a set of concepts $\mathcal{L}(x)$ which the individual must satisfy and each edge is labelled with a role name.

An algorithm starts with a single node labelled $\{D\}$, and proceeds by repeatedly applying a set of *expansion rules* which recursively decompose the concepts in node labels; new edges and nodes are added as required in order to satisfy $\exists R.C$ concepts. The construction terminates either when none of the rules can be applied in a way which extends the tree, or when the discovery of obvious contradictions demonstrates that D has no model.

*Part of this work was carried out while being a guest at IRST, Trento.
†This work was supported by the Esprit Project 22469 – DWQ.
[1] Note that $\mathtt{has\_part}$ is taken to be the inverse of $\mathtt{is\_part}$.

[2] Details that have been omitted in the interests of brevity can be found in [Horrocks & Sattler,1998].

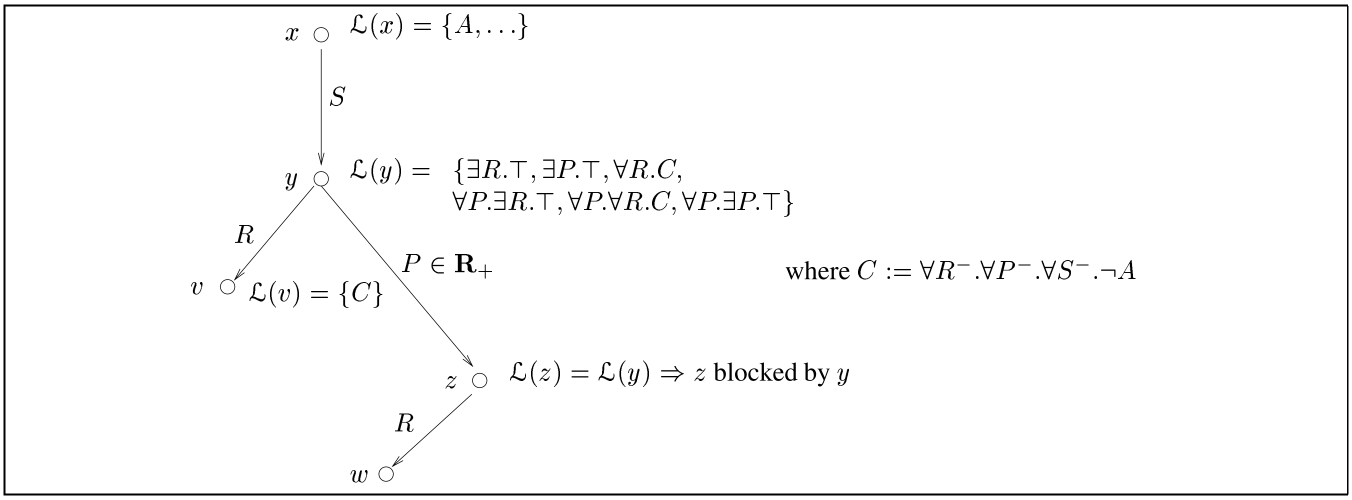

Figure 1: A tableau where dynamic blocking is crucial.

In order to prove that such an algorithm is a sound and complete decision procedure for concept satisfiability in a given logic, it is necessary to demonstrate that the models it constructs are valid with respect to the semantics, that it will always find a model if one exists and that it always terminates. The first two points can usually be dealt with by proving that the expansion rules preserve satisfiability, and that in the case of non-deterministic expansion (e.g., of disjunctions) all possibilities are exhaustively searched. For logics such as $\mathcal{ALC}$, termination is mainly due to the fact that the expansion rules can only add new concepts which are strictly smaller than the decomposed concept, so the model must stabilise when all concepts have been fully decomposed.

Termination is not, however, guaranteed for logics which include transitive roles, as the expansion rules can introduce new concepts which are the same size as the decomposed concept. In particular, $\forall R.C$ concepts, where R is a transitive role, are dealt with by propagating the whole concept across R labelled edges. For example, given a leaf node x labelled $\{C, \exists R.C, \forall R.(\exists R.C)\}$, where R is a transitive role, the combination of the $\exists R.C$ and $\forall R.(\exists R.C)$ concepts would cause a new node y to be added to the tree with an identical label to x. The expansion process could then be repeated indefinitely.

This problem can be dealt with by *blocking*: halting the expansion process when a cycle is detected [Baader,1991; Buchheit *et al.*,1993]. For logics without inverse roles, the general procedure is to check the label of each new node y, and if it is a *subset* [Baader *et al.*,1996] of the label of an existing node x, then no further expansion of y is performed: x is said to block y. The resulting tree corresponds to a cyclical model in which y is identified with x.[3] The validity of the cyclical model is an easy consequence of the fact that the $\exists R.C$ concept which y must satisfy must also be satisfied by x, because x's label is a superset of y's. Termination is guaranteed by the fact that all concepts in node labels are ultimately derived from the decomposition of D, so all node labels must be a subset of the subconcepts of D, and a cycle must therefore occur within a finite number of expansion steps.

Blocking is, however, more problematical when inverse roles are added to the logic, and a key feature of the algorithms presented here is the introduction of a *dynamic blocking* strategy using label equality instead of subset. With inverse roles, the blocking condition must be equality of node labels, because roles are now bi-directional and additional concepts in x's label could invalidate the model with respect to y's predecessor. Taking the above example of a node labelled $\{C, \exists R.C, \forall R.(\exists R.C)\}$, if the successor of this node were blocked by a node whose label additionally included $\forall R^-.\neg C$, then the cyclical model would clearly be invalid.

Another difficulty introduced by inverse roles is the fact that it is no longer possible to establish a block on a once and for all basis when a new node is added to the tree, because further expansion in other parts of the tree could lead to the labels of the blocking and/or blocked nodes being extended and the block being invalidated. For example, consider the example sketched in Figure 1. It shows parts of a tableau that was generated for the concept

$$A \sqcap \exists S.(\ \forall P.\exists R.\top \sqcap \forall P.\forall R.C \sqcap \forall P.\exists P.\top\ \ .\\ \exists R.\top \sqcap \exists P.\top \sqcap \forall R.C \sqcap)$$

For C as given in Figure 1, this concept is not satisfiable: w has to be an instance of C, which implies that x is an instance of $\neg A$—which is inconsistent with x being an instance of A.

[3] For logics with a transitive closure operator it is necessary to check the validity of the cyclical model created by blocking [Baader,1991], but for logics which only support transitive roles the cyclical model is always valid [Sattler,1996].

As P is a transitive role, all universal value restrictions over P are propagated from y to z, hence $\mathcal{L}(y) = \mathcal{L}(z)$ and z is blocked by y. Now, if the blocking of z would not be broken when $\forall P^-.\forall S^-.\neg A$ is added to $\mathcal{L}(y)$ from $C \in \mathcal{L}(v)$, then $\neg A$ would be never added to $\mathcal{L}(x)$ and the inconsistency would not be detected.

Moreover, it is necessary to continue with some expansion of blocked nodes, because $\forall R.C$ concepts in their labels could effect other parts on the tree: Again, let us consider the example in Figure 1: After the blocking of z was broken and $\forall P^-.\forall S^-.\neg A$ added to both $\mathcal{L}(y)$ and $\mathcal{L}(z)$, z is again blocked by y. However, the universal value restriction $\forall P^-.\forall S^-.\neg A \in \mathcal{L}(z)$ has to be expanded in order to detect the inconsistency.

This problem is overcome by using dynamic blocking: allowing blocks to be dynamically established and broken as the expansion progresses, and continuing to expand $\forall R.C$ concepts in the labels of blocked nodes.

3 Syntax and Semantics of $\mathcal{ALCI}_{R^+}$

For ease of understanding, we start by introducing the Description Logic $\mathcal{ALCI}_{R^+}$, which is the extension of the well-known DL $\mathcal{ALC}$ [Schmidt-Schauß & Smolka,1988] with *transitively closed roles* and *inverse* (converse) roles. The set of transitive role names $\mathbf{R}_+$ is a subset of the set of role names $\mathbf{R}$. Interpretations map role names to binary relations on the interpretation domain, and transitive role names to transitive relations. In addition, for any role $R \in \mathbf{R}$, the role R^- is interpreted as the inverse of R.

In the next section, we describe a tableaux algorithm for testing the satisfiability of $\mathcal{ALCI}_{R^+}$ concepts and present a proof of its soundness and completeness. The extension of $\mathcal{ALCI}_{R^+}$ by role hierarchies, $\mathcal{ALCHI}_{R^+}$, together with the extended tableaux algorithm and corresponding proofs is then described in Section 5.

Definition 1 Let N_C be a set of *concept names* and let $\mathbf{R}$ be a set of *role names* with transitive role names $\mathbf{R}_+ \subseteq \mathbf{R}$. The set of *$\mathcal{ALCI}_{R^+}$-roles* is $\mathbf{R} \cup \{R^- \mid R \in \mathbf{R}\}$. The set of *$\mathcal{ALCI}_{R^+}$-concepts* is the smallest set such that

1. every concept name is a concept and
2. if C and D are concepts and R is an $\mathcal{ALCI}_{R^+}$-role, then $(C \sqcap D)$, $(C \sqcup D)$, $(\neg C)$, $(\forall R.C)$, and $(\exists R.C)$ are concepts.

An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a set $\Delta^\mathcal{I}$, called the *domain* of $\mathcal{I}$, and a function $\cdot^\mathcal{I}$ which maps every concept to a subset of $\Delta^\mathcal{I}$ and every role to a subset of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$ such that, for all concepts C, D, the properties in Figure 2 are satisfied.

A concept C is called *satisfiable* iff there is some interpretation $\mathcal{I}$ such that $C^\mathcal{I} \neq \emptyset$. Such an interpretation is called a *model of* C. A concept D *subsumes* a concept C (written $C \sqsubseteq D$) iff $C^\mathcal{I} \subseteq D^\mathcal{I}$ holds for each interpretation $\mathcal{I}$. For an interpretation $\mathcal{I}$, an individual $x \in \Delta^\mathcal{I}$ is called an *instance* of a concept C iff $x \in C^\mathcal{I}$.

In order to make the following considerations easier, we introduce two functions on roles:

1. The inverse relation on roles is symmetric, and to avoid considering roles such as R^{--}, we define a function Inv which returns the inverse of a role. More precisely, $\mathsf{Inv}(R) = R^-$ if R is a role name, and $\mathsf{Inv}(R) = S$ if $R = S^-$.
2. Obviously, a role R is transitive if and only if $\mathsf{Inv}(R)$ is transitive. However, this may be established by either R or $\mathsf{Inv}(R)$ being in $\mathbf{R}_+$. We therefore define a function Trans which returns true iff R is a transitive role—regardless of whether it is a role name or the inverse of a role name. More precisely, $\mathsf{Trans}(R) = \mathrm{true}$ iff $R \in \mathbf{R}_+$ or $\mathsf{Inv}(R) \in \mathbf{R}_+$.

4 A Tableaux Algorithm for $\mathcal{ALCI}_{R^+}$

Like other tableaux algorithms, the $\mathcal{ALCI}_{R^+}$ algorithm tries to prove the satisfiability of a concept D by constructing a model of D. The model is represented by a so-called *completion tree*, a tree some of whose nodes correspond to individuals in the model, each node being labelled with a set of $\mathcal{ALCI}_{R^+}$-concepts. When testing the satisfiability of an $\mathcal{ALCI}_{R^+}$-concept D, these sets are restricted to subsets of *sub*(D), where *sub*(D) is the set of subconcepts of D.

For ease of construction, we assume all concepts to be in *negation normal form* (NNF), that is, negation occurs only in front of concept names. Any $\mathcal{ALCI}_{R^+}$-concept can easily be transformed to an equivalent one in NNF by pushing negations inwards.

The soundness and completeness of the algorithm will be proved by showing that it creates a *tableau* for D. We have chosen to take the (not so) long way round tableaux for proving properties of tableaux algorithms because—once tableaux are defined and Lemma 1 is proven—the remaining proofs are considerable easier.

Definition 2 If D is an $\mathcal{ALCI}_{R^+}$-concept in NNF and $\mathbf{R}_D$ is the set of roles occurring in D, together with their inverses, a tableau T for D is defined to be a triple $(\mathbf{S}, \mathcal{L}, \mathcal{E})$ such that: $\mathbf{S}$ is a set of individuals, $\mathcal{L} : \mathbf{S} \to 2^{sub(D)}$ maps each individual to a set of concepts which is a subset of *sub*(D), $\mathcal{E} : \mathbf{R}_D \to 2^{\mathbf{S} \times \mathbf{S}}$ maps each role in $\mathbf{R}_D$ to a set of pairs of individuals, and there is some individual $s \in \mathbf{S}$ such that $D \in \mathcal{L}(s)$. For all $s \in \mathbf{S}$, $C, E \in sub(D)$, and $R \in \mathbf{R}_D$, it holds that:

1. if $C \in \mathcal{L}(s)$, then $\neg C \notin \mathcal{L}(s)$,
2. if $C \sqcap E \in \mathcal{L}(s)$, then $C \in \mathcal{L}(s)$ and $E \in \mathcal{L}(s)$,
3. if $C \sqcup E \in \mathcal{L}(s)$, then $C \in \mathcal{L}(s)$ or $E \in \mathcal{L}(s)$,
4. if $\forall R.C \in \mathcal{L}(s)$ and $\langle s, t\rangle \in \mathcal{E}(R)$, then $C \in \mathcal{L}(t)$,

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}, \qquad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}, \qquad \neg C^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}},$$
$$(\exists S.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \text{There exists } y \in \Delta^{\mathcal{I}} \text{ with } \langle x,y\rangle \in S^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\},$$
$$(\forall S.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \text{For all } y \in \Delta^{\mathcal{I}}, \text{ if } \langle x,y\rangle \in S^{\mathcal{I}}, \text{ then } y \in C^{\mathcal{I}}\},$$
$$\text{For } S \in \mathbf{R}: \quad \langle x,y\rangle \in S^{\mathcal{I}} \text{ iff } \langle y,x\rangle \in S^{-\mathcal{I}}, \text{ and}$$
$$\text{For } R \in \mathbf{R}_+: \quad \text{if } \langle x,y\rangle \in R^{\mathcal{I}} \text{ and } \langle y,z\rangle \in R^{\mathcal{I}}, \text{ then } \langle x,z\rangle \in R^{\mathcal{I}}.$$

Figure 2: Semantics of $\mathcal{ALCI}_{R^+}$-concepts

5. if $\exists R.C \in \mathcal{L}(s)$, then there is some $t \in \mathbf{S}$ such that $\langle s,t\rangle \in \mathcal{E}(R)$ and $C \in \mathcal{L}(t)$,
6. if $\forall R.C \in \mathcal{L}(s)$, $\langle s,t\rangle \in \mathcal{E}(R)$ and $\mathsf{Trans}(R)$, then $\forall R.C \in \mathcal{L}(t)$, and
7. $\langle x,y\rangle \in \mathcal{E}(R)$ iff $\langle y,x\rangle \in \mathcal{E}(\mathsf{Inv}(R))$.

Lemma 1 *An $\mathcal{ALCI}_{R^+}$-concept D is satisfiable iff there exists a tableau for D.*

Proof: For the *if* direction, if $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ is a tableau for D with $D \in \mathcal{L}(s_0)$, a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of D can be defined as:

$$\Delta^{\mathcal{I}} = \mathbf{S}$$

for all concept names A in *sub*(D):

$$A^{\mathcal{I}} = \{s \mid A \in \mathcal{L}(s)\}$$
$$R^{\mathcal{I}} = \begin{cases} \mathcal{E}(R)^+ & \text{if } \mathsf{Trans}(R) \\ \mathcal{E}(R) & \text{otherwise} \end{cases}$$

where $\mathcal{E}(R)^+$ denotes the transitive closure of $\mathcal{E}(R)$. $D^{\mathcal{I}} \neq \emptyset$ because $s_0 \in D^{\mathcal{I}}$. Transitive roles are obviously interpreted as transitive relations. By induction on the structure of concepts, we show that, if $E \in \mathcal{L}(s)$, then $s \in E^{\mathcal{I}}$. Let $E \in \mathcal{L}(s)$.

1. If E is a concept name, then $s \in E^{\mathcal{I}}$ by definition.
2. If $E = \neg C$, then $C \notin \mathcal{L}(s)$ (due to Property 1 in Definition 2), so $s \in \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} = E^{\mathcal{I}}$.
3. If $E = (C_1 \sqcap C_2)$, then $C_1 \in \mathcal{L}(s)$ and $C_2 \in \mathcal{L}(s)$, so by induction $s \in C_1^{\mathcal{I}}$ and $s \in C_2^{\mathcal{I}}$. Hence $s \in (C_1 \sqcap C_2)^{\mathcal{I}}$.
4. The case $E = (C_1 \sqcup C_2)$ is analogous to 3.
5. If $E = (\exists S.C)$, then there is some $t \in \mathbf{S}$ such that $\langle s,t\rangle \in \mathcal{E}(S)$ and $C \in \mathcal{L}(t)$. By definition, $\langle s,t\rangle \in S^{\mathcal{I}}$ and by induction $t \in C^{\mathcal{I}}$. Hence $S \in (\exists S.C)^{\mathcal{I}}$.
6. If $E = (\forall S.C)$ and $\langle s,t\rangle \in S^{\mathcal{I}}$, then either
   (a) $\langle s,t\rangle \in \mathcal{E}(S)$ and $C \in \mathcal{L}(t)$, or
   (b) $\langle s,t\rangle \notin \mathcal{E}(S)$, then there exists a path of length $n \geq 1$ such that $\langle s,s_1\rangle, \langle s_1,s_2\rangle, \ldots, \langle s_n,t\rangle \in \mathcal{E}(S)$. Due to Property 6 in Definition 2, $\forall S.C \in \mathcal{L}(s_i)$ for all $1 \leq i \leq n$, and we have $C \in \mathcal{L}(t)$.

   In both cases, we have by induction $t \in C^{\mathcal{I}}$, hence $s \in (\forall S.C)^{\mathcal{I}}$.

For the converse, if $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a model of D, then a tableau $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ for D can be defined as:

$$\mathbf{S} = \Delta^{\mathcal{I}}$$
$$\mathcal{E}(R) = R^{\mathcal{I}}$$
$$\mathcal{L}(s) = \{C \in sub(D) \mid s \in C^{\mathcal{I}}\}$$

It only remains to demonstrate that T is a tableau for D:

1. T satisfies properties 1–5 in Definition 2 as a direct consequence of the semantics of $\mathcal{ALCI}_{R^+}$ concepts.
2. If $d \in (\forall R.C)^{\mathcal{I}}$, $\langle d,e\rangle \in R^{\mathcal{I}}$ and $\mathsf{Trans}(R)$, then $e \in (\forall R.C)^{\mathcal{I}}$ unless there is some f such that $\langle e,f\rangle \in R^{\mathcal{I}}$ and $f \notin C^{\mathcal{I}}$. However, if $\langle d,e\rangle \in R^{\mathcal{I}}$, $\langle e,f\rangle \in R^{\mathcal{I}}$ and $R \in \mathbf{R}_+$, then $\langle d,f\rangle \in R^{\mathcal{I}}$ and $d \notin (\forall R.C)^{\mathcal{I}}$. T therefore satisfies Property 6 in Definition 2.
3. T satisfies Property 7 in Definition 2 as a direct consequence of the semantics of inverse relations. ∎

4.1 Constructing an $\mathcal{ALCI}_{R^+}$ Tableau

From Lemma 1, an algorithm which constructs a tableau for an $\mathcal{ALCI}_{R^+}$-concept D can be used as a decision procedure for the satisfiability of D. Such an algorithm will now be described in detail.

The tableaux algorithm works on *completion trees*. This is a tree where each node x of the tree is labelled with a set $\mathcal{L}(x) \subseteq sub(D)$ and each edge $\langle x,y\rangle$ is labelled $\mathcal{L}(\langle x,y\rangle) = R$ for some (possibly inverse) role R occurring in *sub*(D). Edges are added when expanding $\exists R.C$ and $\exists R^-.C$ terms; they correspond to relationships between pairs of individuals and are always directed from the root node to the leaf nodes. The algorithm expands the tree either by extending $\mathcal{L}(x)$ for some node x or by adding new leaf nodes.

For a node x, $\mathcal{L}(x)$ is said to contain a *clash* if, for some concept C, $\{C, \neg C\} \subseteq \mathcal{L}(x)$.

If nodes x and y are connected by an edge $\langle x,y\rangle$, then y is called a *successor* of x and x is called a *predecessor* of y; *ancestor* is the transitive closure of *predecessor*.

A node y is called an *R-neighbour* of a node x if either y is a successor of x and $\mathcal{L}(\langle x,y\rangle) = R$ or y is a predecessor of x and $\mathcal{L}(\langle y,x\rangle) = \mathsf{Inv}(R)$.

A node x is *blocked* if for some ancestor y, y is blocked or $\mathcal{L}(x) = \mathcal{L}(y)$. A blocked node x is *indirectly blocked* if its predecessor is blocked, otherwise it is *directly blocked*. If x is directly blocked, it has a unique ancestor y such that $\mathcal{L}(x) = \mathcal{L}(y)$: if there existed another ancestor z such that $\mathcal{L}(x) = \mathcal{L}(z)$ then either y or z must be blocked. If x is directly blocked and y is the unique ancestor such that $\mathcal{L}(x) = \mathcal{L}(y)$, we will say that *$y$ blocks x*.

The algorithm initialises a tree $\mathbf{T}$ to contain a single node x_0, called the *root* node, with $\mathcal{L}(x_0) = \{D\}$, where D is the concept to be tested for satisfiability. $\mathbf{T}$ is then expanded by repeatedly applying the rules from Figure 3.

The completion tree is *complete* when for some node x, $\mathcal{L}(x)$ contains a clash or when none of the rules is applicable. If, for an input concept D, the expansion rules can be applied in such a way that they yield a complete, clash-free completion tree, then the algorithm returns "D is *satisfiable*", and "D is *unsatisfiable*" otherwise.

4.2 Soundness and Completeness

The soundness and completeness of the algorithm will be demonstrated by proving that, for an $\mathcal{ALCI}_{R^+}$-concept D, it always terminates and that it returns *satisfiable* if and only if D is satisfiable.

Lemma 2 *For each $\mathcal{ALCI}_{R^+}$-concept D, the tableaux algorithm terminates.*

Proof: Let $m = |sub(D)|$. Obviously, m is linear in the length of D. Termination is a consequence of the following properties of the expansion rules:

1. The expansion rules never remove nodes from the tree or concepts from node labels.
2. Successors are only generated for existential value restrictions (concepts of the form $\exists R.C$), and for any node each of these restrictions triggers the generation of at most one successor. Since $sub(D)$ contains at most m existential value restrictions, the out-degree of the tree is bounded by m.
3. Nodes are labelled with nonempty subsets of $sub(D)$. If a path p is of length at least 2^m, then there are 2 nodes x, y on p, with $\mathcal{L}(x) = \mathcal{L}(y)$, and blocking occurs. Since a path on which nodes are blocked cannot become longer, paths are of length at most 2^m. ∎

Together with Lemma 1, the following lemma implies soundness of the tableaux algorithm.

Lemma 3 *If the expansion rules can be applied to an $\mathcal{ALCI}_{R^+}$-concept D such that they yield a complete and clash-free completion tree, then D has a tableau.*

Proof: Let $\mathbf{T}$ be the complete and clash-free tree constructed by the tableaux algorithm for D. A tableau $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ can be defined with:

$$\begin{aligned} \mathbf{S} &= \{x \mid x \text{ is a node in } \mathbf{T} \text{ that is not blocked}\}, \\ \mathcal{E}(R) &= \{\langle x, y\rangle \in \mathbf{S} \times \mathbf{S} \mid \\ &\quad 1.\ y \text{ is an } R\text{-neighbour of } x \quad \textit{or} \\ &\quad 2.\ \mathcal{L}(\langle x, z\rangle) = R \text{ and } y \text{ blocks } z \quad \textit{or} \\ &\quad 3.\ \mathcal{L}(\langle y, z\rangle) = \mathsf{Inv}(R) \text{ and } x \text{ blocks } z\}, \end{aligned}$$

and it can be shown that T is a tableau for D:

1. $D \in \mathcal{L}(x_0)$ for the root x_0 of $\mathbf{T}$ and, as x_0 has no predecessors, it cannot be blocked. Hence $D \in \mathcal{L}(s)$ for some $s \in \mathbf{S}$.
2. Property 1 of Definition 2 is satisfied because $\mathbf{T}$ is clash-free.
3. Properties 2 and 3 of Definition 2 are satisfied because neither the $\sqcap$-rule nor the $\sqcup$-rule apply to any $x \in \mathbf{S}$.
4. Property 4 in Definition 2 is satisfied because for all $x \in \mathbf{S}$, if $\forall R.C \in \mathcal{L}(x)$ and $\langle x, y\rangle \in \mathcal{E}(R)$ then either:
   (a) x is an R-neighbour of y,
   (b) $\mathcal{L}(\langle x, z\rangle) = R$, y blocks z, from the $\forall$-rule $C \in \mathcal{L}(z)$, $\mathcal{L}(y) = \mathcal{L}(z)$, or
   (c) $\mathcal{L}(\langle y, z\rangle) = \mathsf{Inv}(R)$, x blocks z, $\mathcal{L}(x) = \mathcal{L}(z)$, so from the $\forall$-rule $C \in \mathcal{L}(y)$.

   In all 3 cases, the $\forall$-rule ensures that $C \in \mathcal{L}(y)$.
5. Property 5 in Definition 2 is satisfied because for all $x \in \mathbf{S}$, if $\exists R.C \in \mathcal{L}(x)$, then the $\exists$-rule ensures that there is either:
   (a) a predecessor y such that $\mathcal{L}(\langle y, x\rangle) = \mathsf{Inv}(R)$ and $C \in \mathcal{L}(y)$. Because y is a predecessor of x it cannot be blocked, so $y \in \mathbf{S}$ and $\langle y, x\rangle \in \mathcal{E}(R)$.
   (b) a successor y such that $\mathcal{L}(\langle x, y\rangle) = R$ and $C \in \mathcal{L}(y)$. If y is not blocked, then $y \in \mathbf{S}$ and $\langle x, y\rangle \in \mathcal{E}(R)$. Otherwise, y is blocked by some z with $\mathcal{L}(z) = \mathcal{L}(y)$. Hence $C \in \mathcal{L}(z)$, $z \in \mathbf{S}$ and $\langle x, z\rangle \in \mathcal{E}(R)$.
6. Property 6 in Definition 2 is satisfied because for all $x \in \mathbf{S}$, if $\forall R.C \in \mathcal{L}(x)$, $\langle x, y\rangle \in \mathcal{E}(R)$, and $\mathsf{Trans}(R)$, then either:
   (a) x is an R-neighbour of y,
   (b) $\mathcal{L}(\langle x, z\rangle) = R$, y blocks z, and $\mathcal{L}(y) = \mathcal{L}(z)$, or
   (c) $\mathcal{L}(\langle y, z\rangle) = \mathsf{Inv}(R)$, x blocks z, hence $\mathcal{L}(x) = \mathcal{L}(z)$ and $\forall R.C \in \mathcal{L}(z)$.

   In all 3 cases, the $\forall_+$-rule ensures that $\forall R.C \in \mathcal{L}(y)$.
7. Property 7 in Definition 2 is satisfied because for each $\langle x, y\rangle \in \mathcal{E}(R)$, either:
   (a) x is an R-neighbour of y, so y is an $\mathsf{Inv}(R)$-neighbour of x and $\langle y, x\rangle \in \mathcal{E}(\mathsf{Inv}(R))$.
   (b) $\mathcal{L}(\langle x, z\rangle) = R$ and y blocks z, so $\mathcal{L}(\langle x, z\rangle) = \mathsf{Inv}(\mathsf{Inv}(R))$ and $\langle y, x\rangle \in \mathcal{E}(\mathsf{Inv}(R))$.
   (c) $\mathcal{L}(\langle y, z\rangle) = \mathsf{Inv}(R)$ and x blocks z, so $\langle y, x\rangle \in \mathcal{E}(\mathsf{Inv}(R))$. ∎

$\sqcap$-rule:	if 1. $C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and 2. $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$		then $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
$\sqcup$-rule:	if 1. $C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and 2. $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$	then, for some $C \in \{C_1, C_2\}$,	$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C\}$
$\exists$-rule:	if 1. $\exists S.C \in \mathcal{L}(x)$, x is not blocked, and 2. x has no S-neighbour y with $C \in \mathcal{L}(y)$:		then create a new node y with $\mathcal{L}(\langle x, y\rangle) = S$ and $\mathcal{L}(y) = \{C\}$
$\forall$-rule:	if 1. $\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked, and 2. there is an S-neighbour y of x with $C \notin \mathcal{L}(y)$		then $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{C\}$
$\forall_+$-rule:	if 1. $\forall S.C \in \mathcal{L}(x)$, $\mathsf{Trans}(S)$, x is not indirectly blocked, and 2. there is an S-neighbour y of x with $\forall S.C \notin \mathcal{L}(y)$		then $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{\forall S.C\}$

Figure 3: Tableaux expansion rules for $\mathcal{ALCI}_{R^+}$

Lemma 4 *If D has a tableau, then the expansion rules can be applied in such a way that the tableaux algorithm yields a complete and clash-free completion tree for D.*

Proof: Let $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ be a tableau for D. Using T, we trigger the application of the expansion rules such that they yield a completion tree $\mathbf{T}$ that is both complete and clash-free. We start with $\mathbf{T}$ consisting of a single node x_0, the root, with $\mathcal{L}(x_0) = \{D\}$.

T is a tableau, hence there is some $s_0 \in \mathbf{S}$ with $D \in \mathcal{L}(s_0)$. When applying the expansion rules to $\mathbf{T}$, the application of the non-deterministic $\sqcup$-rule is driven by the labelling in the tableau T. To this purpose, we define a mapping π which maps the nodes of $\mathbf{T}$ to elements of $\mathbf{S}$, and we steer the application of the $\sqcup$-rule such that $\mathcal{L}(x) \subseteq \mathcal{L}(\pi(x))$ holds for all nodes x of the completion tree.

More precisely, we define π inductively as follows:

- $\pi(x_0) = s_0$.
- If $\pi(x_i) = s_i$ is already defined, and a successor y of x_i was generated for $\exists R.C \in \mathcal{L}(x_i)$, then $\pi(y) = t$ for some $t \in \mathbf{S}$ with $C \in \mathcal{L}(t)$ and $\langle s_i, t\rangle \in \mathcal{E}(R)$.

To make sure that we have $\mathcal{L}(x_i) \subseteq \mathcal{L}(\pi(x_i))$, we use the $\sqcup'$-rule given in Figure 4 instead of the $\sqcup$-rule.

$\sqcup'$-rule: if 1. $C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and 2. $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$ then $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C\}$ for some $C \in \{C_1, C_2\} \cap \mathcal{L}(\pi(x))$

Figure 4: The $\sqcup'$-rule

The expansion rules given in Figure 3 with the $\sqcup$-rule replaced by the $\sqcup'$-rule are called *modified* expansion rules in the following.

It is easy to see that, if a tree $\mathbf{T}$ was generated using the modified expansion rules, then the expansion rules can be applied in such a way that they yield $\mathbf{T}$. Hence Lemma 3 and Lemma 2 still apply, and thus using the $\sqcup'$-rule instead of the $\sqcup$-rule preserves soundness and termination.

We will now show by induction that, if $\mathcal{L}(x) \subseteq \mathcal{L}(\pi(x))$ holds for all nodes x in $\mathbf{T}$, then the application of an expansion rule preserves this subset-relation. To start with, we clearly have $\{D\} = \mathcal{L}(x_0) \subseteq \mathcal{L}(s_0)$.

If the $\sqcap$-rule can be applied to x in $\mathbf{T}$ with $C = C_1 \sqcap C_2 \in \mathcal{L}(x)$, then C_1, C_2 are added to $\mathcal{L}(x)$. Since T is a tableau, $\{C_1, C_2\} \subseteq \mathcal{L}(\pi(x))$, and hence the $\sqcap$-rule preserves $\mathcal{L}(x) \subseteq \mathcal{L}(\pi(x))$.

If the $\sqcup'$-rule can be applied to x in $\mathbf{T}$ with $C = C_1 \sqcup C_2 \in \mathcal{L}(x)$, then $C \in \{C_1, C_2\}$ is in $\mathcal{L}(\pi(x))$, and C is added to $\mathcal{L}(x)$ by the $\sqcup'$-rule. Hence the $\sqcup'$-rule preserves $\mathcal{L}(x) \subseteq \mathcal{L}(\pi(x))$.

If the $\exists$-rule can be applied to x in $\mathbf{T}$ with $C = \exists R.C_1 \in \mathcal{L}(x)$, then $C \in \mathcal{L}(\pi(x))$ and there is some $t \in \mathbf{S}$ with $\langle \pi(x), t\rangle \in \mathcal{E}(R)$ and $C_1 \in \mathcal{L}(t)$. The $\exists$-rule creates a new successor y of x for which $\pi(y) = t$ for some t with $C_1 \in \mathcal{L}(t)$. Hence we have $\mathcal{L}(y) = \{C_1\} \subseteq \mathcal{L}(\pi(y))$.

If the $\forall$-rule can be applied to x in $\mathbf{T}$ with $C = \forall R.C_1 \in \mathcal{L}(x)$ and y is an R-neighbour of x, then $\langle \pi(x), \pi(y)\rangle \in \mathcal{E}(R)$, and thus $C_1 \in \mathcal{L}(\pi(y))$. The $\forall$-rule adds C_1 to $\mathcal{L}(y)$ and thus preserves $\mathcal{L}(x) \subseteq \mathcal{L}(\pi(x))$.

If the $\forall_+$-rule can be applied to x in $\mathbf{T}$ with $C = \forall R.C_1 \in \mathcal{L}(x)$, $\mathsf{Trans}(R)$ and y being an R-neighbour of x, then $\langle \pi(x), \pi(y)\rangle \in \mathcal{E}(R)$, and thus $\forall R.C_1 \in \mathcal{L}(\pi(y))$. The $\forall_+$-rule adds C_1 to $\mathcal{L}(y)$ and thus preserves $\mathcal{L}(y) \subseteq \mathcal{L}(\pi(y))$.

Summing up, the tableau-construction triggered by T terminates with a complete tree, and since $\mathcal{L}(x) \subseteq \mathcal{L}(\pi(x))$ holds for all nodes x in $\mathbf{T}$, $\mathbf{T}$ is clash-free due to Property 1 of Definition 2. ∎

Theorem 1 *The tableaux algorithm is a decision procedure for the satisfiability and subsumption of $\mathcal{ALCI}_{R^+}$-concepts.*

Theorem 1 is an immediate consequence of the Lemmata 1, 2, 3, and 4. Moreover, since $\mathcal{ALCI}_{R^+}$ is closed under negation, subsumption $C \sqsubseteq D$ can be reduced to unsatisfiability of $C \sqcap \neg D$.

5 $\mathcal{ALCI}_{R^+}$ Extended by Role Hierarchies

We will now extend the tableaux algorithm presented in Section 4.1 to deal with *role hierarchies* in a similar way to the algorithm for $\mathcal{ALCH}_{R^+}$ presented in [Horrocks & Gough,1997]. $\mathcal{ALCHI}_{R^+}$ extends $\mathcal{ALCI}_{R^+}$ by allowing, additionally, for inclusion axioms on roles. These axioms can involve transitive as well as non-transitive roles, and inverse roles as well as role names. For example, to express that a role R is symmetric, we add the two axioms $R \sqsubseteq R^-$ and $R^- \sqsubseteq R$.

Definition 3 A *role inclusion axiom* is of the form

$$R \sqsubseteq S,$$

for two (possibly inverse) roles R and S.

For a set of role inclusion axioms $\mathcal{R}$, $\mathcal{R}^+ := (\mathcal{R} \cup \{\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \mid R \sqsubseteq S \in \mathcal{R}\}, \underline{\circledast})$ is called a *role hierarchy*, where $\underline{\circledast}$ is the transitive-reflexive closure of $\sqsubseteq$ over $\mathcal{R} \cup \{\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \mid R \sqsubseteq S \in \mathcal{R}\}$.

$\mathcal{ALCHI}_{R^+}$ is the extension of $\mathcal{ALCI}_{R^+}$ obtained by allowing, additionally, for a role hierarchy $\mathcal{R}^+$.

As well as being correct for $\mathcal{ALCI}_{R^+}$ concepts, an $\mathcal{ALCHI}_{R^+}$ interpretation has to satisfy, for all roles R, S with $R \,\underline{\circledast}\, S$, the additional condition

$$\langle x, y\rangle \in R^{\mathcal{I}} \text{ implies } \langle x, y\rangle \in S^{\mathcal{I}}.$$

The tableaux algorithm given in the preceding section can easily be modified to decide satisfiability of $\mathcal{ALCHI}_{R^+}$-concepts by extending the definitions of both R-neighbours and the $\forall_+$-rule to include the notion of role hierarchies. To prove the soundness and correctness of the extended algorithm, the definition of a tableau is also extended.

Definition 4 As well as satisfying Definition 2 (i.e., being a valid $\mathcal{ALCI}_{R^+}$ tableau), a tableau $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ for an $\mathcal{ALCHI}_{R^+}$-concept D must also satisfy:

6′. if $\forall S.C \in \mathcal{L}(s)$ and $\langle s, t\rangle \in \mathcal{E}(R)$ for some $R \,\underline{\circledast}\, S$ with $\mathsf{Trans}(R)$, then $\forall R.C \in \mathcal{L}(t)$,

8. if $\langle x, y\rangle \in \mathcal{E}(R)$ and $R \,\underline{\circledast}\, S$, then $\langle x, y\rangle \in \mathcal{E}(S)$,

where Property 6′ extends and supersedes Property 6 from Definition 2.

For the $\mathcal{ALCHI}_{R^+}$ algorithm, the $\forall_+$-rule is replaced with the $\forall'_+$-rule (see Figure 5) and the definition of R-neighbours is extended as follows:

Definition 5 Given a completion tree, a node y is called an *R-neighbour* of a node x if either y is a successor of x and $\mathcal{L}(\langle x, y\rangle) = S$ or y is a predecessor of x and $\mathcal{L}(\langle y, x\rangle) = \mathsf{Inv}(S)$ for some S with $S \,\underline{\circledast}\, R$.

In the following, the tableaux algorithm resulting from these modifications will be called the *modified tableaux algorithm*. Due to this definition and the reflexivity of $\underline{\circledast}$, the $\forall'_+$-rule extends the $\forall_+$-rule.

To prove that the modified tableaux algorithm is indeed a decision procedure for the satisfiability of $\mathcal{ALCHI}_{R^+}$-concepts, all 4 technical lemmata used in Section 4.2 to prove this fact for the $\mathcal{ALCI}_{R^+}$ tableaux algorithm have to be re-proven for $\mathcal{ALCHI}_{R^+}$. In the following, we will restrict our attention to cases that differ from those already considered for $\mathcal{ALCI}_{R^+}$.

Lemma 5 *An $\mathcal{ALCHI}_{R^+}$-concept D is satisfiable iff there exists a tableau for D.*

Proof: For the *if* direction, the construction of a model of D from a tableau for D is similar to the one presented in the proof of Lemma 1. If $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ is a tableau for D with $D \in \mathcal{L}(s_0)$, a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of D can be defined as follows:

$$\Delta^{\mathcal{I}} = \mathbf{S}$$

for all concept names A in *sub*(D)

$$A^{\mathcal{I}} = \{s \mid \mathsf{A} \in \mathcal{L}(s)\}$$

$$R^{\mathcal{I}} = \begin{cases} \mathcal{E}(R)^+ & \text{if } \mathsf{Trans}(R) \\ \mathcal{E}(R) \cup \bigcup\limits_{P \,\underline{\circledast}\, R, P \neq R} P^{\mathcal{I}} & \text{otherwise} \end{cases}$$

The interpretation of non-transitive roles is recursive in order to correctly interpret those non-transitive roles that have a transitive sub-role. From the definition of $R^{\mathcal{I}}$ and Property 8 of a tableau it follows that if $\langle x, y\rangle \in S^{\mathcal{I}}$, then either $\langle x, y\rangle \in \mathcal{E}(S)$ or there exists a path $\langle s, s_1\rangle, \langle s_1, s_2\rangle, \ldots, \langle s_n, t\rangle \in \mathcal{E}(R)$ for some R with $\mathsf{Trans}(R)$ and $R \,\underline{\circledast}\, S$.

Property 8 of a tableau ensures that $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ holds for all roles with $R \,\underline{\circledast}\, S$, including those cases where R is a transitive role. Again, it can be shown by induction on the structure of concepts that $\mathcal{I}$ is a correct interpretation. We restrict our attention to the only case that is different from the ones in the proof of Lemma 1. Let $E \in$ *sub*(D).

6′. If $E = (\forall S.C)$ and $\langle s, t\rangle \in S^{\mathcal{I}}$, then either

(a) $\langle s, t\rangle \in \mathcal{E}(S)$ and $C \in \mathcal{L}(t)$, or

(b) $\langle s, t\rangle \notin \mathcal{E}(S)$, then there exists a path of length $n \geq 1$ such that $\langle s, s_1\rangle, \langle s_1, s_2\rangle, \ldots, \langle s_n, t\rangle \in \mathcal{E}(R)$ for some R with $\mathsf{Trans}(R)$ and $R \,\underline{\circledast}\, S$. Due to Property 6′, $\forall R.C \in \mathcal{L}(s_i)$ for all $1 \leq i \leq n$, and we have $C \in \mathcal{L}(t)$.

In both cases, we have $t \in C^{\mathcal{I}}$.

$\forall'_+$-rule:	if 1.	$\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked, and
	2.	there is some R with $\mathsf{Trans}(R)$ and $R \sqsubseteq^* S$,
	3.	there is an R-neighbour y of x with $\forall R.C \notin \mathcal{L}(y)$ then $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{\forall R.C\}$

Figure 5: The new $\forall_+$-rule for $\mathcal{ALCHI}_{R^+}$.

For the converse, if $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a model of D, then a tableau $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ for D is defined like the one defined in the proof of Lemma 1.

It remains to demonstrate that T is a tableau for D:

1. T satisfies properties 1–5 in Definition 2 as a direct consequence of the semantics of $\mathcal{ALCHI}_{R^+}$-concepts.
2. If $d \in (\forall S.C)^{\mathcal{I}}$ and $\langle d, e\rangle \in R^{\mathcal{I}}$ for R with $\mathsf{Trans}(R)$ and $R \sqsubseteq^* S$, then $e \in (\forall R.C)^{\mathcal{I}}$ unless there is some f such that $\langle e, f\rangle \in R^{\mathcal{I}}$ and $f \notin C^{\mathcal{I}}$. In this case, if $\langle d, e\rangle \in R^{\mathcal{I}}$, $\langle e, f\rangle \in R^{\mathcal{I}}$ and $\mathsf{Trans}(R)$, then $\langle d, f\rangle \in R^{\mathcal{I}}$. Hence $\langle d, f\rangle \in S^{\mathcal{I}}$ and $d \notin (\forall S.C)^{\mathcal{I}}$—in contradiction of the assumption. T therefore satisfies Property $6'$ in Definition 4.
3. Since $\mathcal{I}$ is a model of D, $\langle x, y\rangle \in R^{\mathcal{I}}$ implies $\langle x, y\rangle \in S^{\mathcal{I}}$ for all roles R, S with $R \sqsubseteq^* S$. Hence T satisfies Property 8 in Definition 4. ∎

Lemma 6 *For each $\mathcal{ALCHI}_{R^+}$-concept D, the modified tableaux algorithm terminates.*

Proof: Identical to the one given for Lemma 2.

Lemma 7 *If the expansion rules can be applied to an $\mathcal{ALCHI}_{R^+}$-concept D such that they yield a complete and clash-free completion tree, then D has a tableau.*

Proof: The definition of a tableau from a complete and clash-free completion tree, as presented in the proof of Lemma 3, has to be slightly modified. A tableau $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ is now defined with:

$$
\begin{aligned}
\mathbf{S} &= \{x \mid x \text{ is a node in } \mathbf{T} \text{ that is not blocked}\} \\
\mathcal{E}(S) &= \{\langle x, y\rangle \in \mathbf{S} \times \mathbf{S} \mid \\
&\quad 1.\ y \text{ is an } S\text{-neighbour of } x \quad \text{or} \\
&\quad 2.\ \text{There exists a role } R \text{ with } R \sqsubseteq^* S \text{ and} \\
&\quad\quad a.\ \mathcal{L}(\langle x, z\rangle) = R \text{ and } y \text{ blocks } z \quad \text{or} \\
&\quad\quad b.\ \mathcal{L}(\langle y, z\rangle) = \mathsf{Inv}(R) \text{ and } x \text{ blocks } z\}
\end{aligned}
$$

and, again, it is shown that T is a tableau for D:

1. Since the expansion rules were started with $\mathcal{L}(x_0) = \{D\}$, $D \in \mathcal{L}(x_0)$ for some $x_0 \in \mathbf{S}$.
2. Properties 1-3, 5 and 7 in Definition 2 are identical to the proof of Lemma 3.
3. Property 4 in Definition 2 is satisfied because for all $x \in \mathbf{S}$, if $\forall S.C \in \mathcal{L}(x)$ and $\langle x, y\rangle \in \mathcal{E}(S)$ then either:
   (a) x is an S-neighbour of y,
   (b) for some role with $R \sqsubseteq^* S$, either
      i. $\mathcal{L}(\langle x, z\rangle) = R$, y blocks z, hence from the $\forall$-rule $C \in \mathcal{L}(z)$, and $\mathcal{L}(y) = \mathcal{L}(z)$, or
      ii. $\mathcal{L}(\langle y, z\rangle) = \mathsf{Inv}(R)$, x blocks z, hence $\mathcal{L}(x) = \mathcal{L}(z)$ and therefor $\forall S.C \in \mathcal{L}(z)$.

   In all cases, the $\forall$-rule ensures $C \in \mathcal{L}(y)$.
4. Property 6' in Definition 4 is satisfied because for all $x \in \mathbf{S}$, if $\forall S.C \in \mathcal{L}(x)$, $\langle x, y\rangle \in \mathcal{E}(R)$ for some R with $\mathsf{Trans}(R)$ and $R \sqsubseteq^* S$, then either:
   (a) y is an R-neighbour of x, or
   (b) there is some role R' with $R' \sqsubseteq R$ and
      i. $\mathcal{L}(\langle x, z\rangle) = R'$, y blocks z and $\mathcal{L}(y) = \mathcal{L}(z)$, or
      ii. $\mathcal{L}(\langle y, z\rangle) = \mathsf{Inv}(R)$, x blocks z and $\mathcal{L}(x) = \mathcal{L}(z)$, hence $\forall S.C \in \mathcal{L}(z)$.

   In all three cases, $\forall R.C \in \mathcal{L}(y)$ follows from the $\forall'_+$-rule.
5. Property 8 in Definition 4 follows immediately from the definition of $\mathcal{E}$. ∎

Lemma 8 *If $\mathcal{ALCHI}_{R^+}$-concept D has a tableau, then the expansion rules can be applied in such a way that the tableaux algorithm yields a complete and clash-free completion tree for D.*

The proof of Lemma 8 is identical to the one presented for Lemma 4. Again, summing up, we have the following theorem.

Theorem 2 *The modified tableaux algorithm is a decision procedure for the satisfiability and subsumption of $\mathcal{ALCHI}_{R^+}$-concepts.*

5.1 General Concept Inclusion Axioms

In [Baader,1991; Schild,1991; Baader *et al.*,1993], the *internalisation* of terminological axioms is introduced. This technique is used to reduce reasoning with respect to a (possibly cyclic) *terminology* to satisfiability of concepts. In [Horrocks & Gough,1997], we saw how role hierarchies can be used to reduce satisfiability and subsumption with respect to a terminology to concept satisfiability and subsumption. In the presence of inverse roles, this reduction must be slightly modified.

Definition 6 A *terminology* $\mathcal{T}$ is a finite set of general concept inclusion axioms,

$$\mathcal{T} = \{C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n\},$$

where C_i, D_i are arbitrary $\mathcal{ALCHI}_{R^+}$-concepts. An interpretation $\mathcal{I}$ is said to be a model of $\mathcal{T}$ iff $C_i^{\mathcal{I}} \subseteq D_i^{\mathcal{I}}$ holds for all $C_i \sqsubseteq D_i \in \mathcal{T}$. C is satisfiable with respect to $\mathcal{T}$ iff there is a model $\mathcal{I}$ of $\mathcal{T}$ with $C^{\mathcal{I}} \neq \emptyset$. Finally, D subsumes C with respect to $\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$) iff for each model $\mathcal{I}$ of $\mathcal{T}$ we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

The following lemma shows how general concept inclusion axioms can be *internalised* using a "universal" role U. This role U is a transitive super-role of all relevant roles and their respective inverses. Hence, for each interpretation $\mathcal{I}$, each individual t reachable via some role path from another individual s is an $U^{\mathcal{I}}$-successor of s. All general concept inclusion axioms $C_i \sqsubseteq D_i$ in $\mathcal{T}$ are propagated along all role paths using the value restriction $\forall U.\neg C \sqcup D$.

Lemma 9 *Let $\mathcal{T}$ be terminology and C, D be $\mathcal{ALCHI}_{R^+}$-concepts and let*

$$C_{\mathcal{T}} := \bigsqcap_{C_i \sqsubseteq D_i \in \mathcal{T}} \neg C_i \sqcup D_i.$$

Let U be a transitive role with $R \sqsubseteq U$, $\mathsf{Inv}(R) \sqsubseteq U$ for each role R that occurs in $\mathcal{T}, C$, or D.

Then C is satisfiable with respect to $\mathcal{T}$ iff

$$C \sqcap C_{\mathcal{T}} \sqcap \forall U.C_{\mathcal{T}}$$

is satisfiable. D subsumes C with respect to $\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$) iff

$$C \sqcap \neg D \sqcap C_{\mathcal{T}} \sqcap \forall U.C_{\mathcal{T}}$$

is unsatisfiable.

Remark: Instead of defining U as a transitive super-role of all roles and their respective inverses, one could have defined U as a transitive super-role of all roles and, additionally, a symmetric role by adding $U \sqsubseteq U^-$ and $U^- \sqsubseteq U$.

The proof of Lemma 9 is similar to the ones that can be found in [Schild,1991; Baader,1990]. One point to show is that, if an $\mathcal{ALCHI}_{R^+}$-concept C is satisfiable with respect to a terminology $\mathcal{T}$, then $C, \mathcal{T}$ have a *connected* model, namely one whose individuals are all related to each other by some role path. This follows from the definition of the semantics of $\mathcal{ALCHI}_{R^+}$-concepts. The other point to proof is that, if y is reachable from x via a role path (possibly involving inverse roles), then $\langle x, y\rangle \in U^{\mathcal{I}}$, which is an easy consequence of the definition of U.

Decidability of satisfiability and subsumption with respect to a terminology is an immediate consequence of Lemma 9 and Theorem 2.

Theorem 3 *The modified tableaux algorithm is a decision procedure for satisfiability and subsumption of $\mathcal{ALCHI}_{R^+}$-concepts with respect to terminologies.*

6 Future work

We intend to extend the logic with functional roles. Functional roles are useful, not only for the representation of aggregated objects, but also in general because they provide a weak form of number restrictions.

The combination of a role hierarchy with transitive, converse and functional roles adds a further level of complexity because satisfiable concepts are no longer guaranteed to have (possibly cyclical) finite models [Schild,1991]. An example of such a concept is:

$$\neg C \sqcap \exists F^-.C \sqcap \forall R^-.(\exists F^-.C)$$

where F is functional, R is transitive and F is a sub-role of R. Any model of this concept must have an infinite sequence of F^- successors, each satisfying C and $\exists F^-.C$, the $\exists F^-.C$ term being propagated along the sequence by the transitive super-role R. Attempting to terminate the sequence in a cycle causes the model to collapse into a single node due the functionality of F, and this leads to an obvious contradiction as the node label will contain both C and $\neg C$.

This problem can be overcome by "unravelling" cyclical models to generate infinite models in which blocked nodes are replaced by copies of the blocking node and its sub-tree. However, to guarantee that local correctness is preserved by the copying process, both the predecessor of a blocked node and the role which connects it to its predecessor, must be the same as those of the blocking node. If this is not the case, then concepts in the label of the blocking node which were satisfied by its predecessor may no longer be satisfied when it is copied onto the blocked node. To ensure that this condition is met, a further enhancement to the blocking strategy must be introduced. Instead of considering single nodes, the enhanced strategy, called pair-wise blocking, considers pairs of nodes and the role which connects them, only establishing a block when a matching node-role-node pattern is found.

The complexity of satisfiability and subsumption of these new extensions of $\mathcal{ALCH}_{R^+}$ is another open problem. From results in [Sattler,1996], it follows that these problems are $\mathsf{ExpTime}$-hard for $\mathcal{ALCHI}_{R^+}$ and PSpace-complete for $\mathcal{ALC}_{R^+}$. Whether these problems are still in PSpace for $\mathcal{ALCI}_{R^+}$ is an open question.

Although $\mathcal{ALCHI}_{R^+}$ and $\mathcal{ALC}$ with the transitive closure and inverse roles are both $\mathsf{ExpTime}$-hard, there are two hints why $\mathcal{ALCHI}_{R^+}$ should have better computational properties than $\mathcal{ALC}$ with the transitive closure and inverse roles: First, the tableaux algorithm for $\mathcal{ALCHI}_{R^+}$ does not have an equivalent of the cut rule—a rule which is strongly responsible for the bad computational behaviour of $\mathcal{ALC}$ with the transitive closure and inverse roles. Intuitively, when a new node x is generated for some $\exists R.C$ concept, this rule non-deterministically chooses a set of concepts from the subconcepts of the initial concept and adds this set to $\mathcal{L}(x)$. We can think of this set as consisting of those concepts which are possibly added to the labelling of x due to universal value restrictions on successors of x. As this set is chosen non-deterministically from exponentially many possibilities, it should be clear that this rule leads to a bad computational behaviour.

Second, the implementation of $\mathcal{ALCH}_{R^+}$ in FaCT [Horrocks & Gough,1997] behaves quite well in realistic application—even though $\mathcal{ALCH}_{R^+}$, too, is ExpTime-hard. Furthermore, this algorithm is amenable to a range of optimisation techniques. We hope that both, this good behaviour and the optimisation techniques, carry over to $\mathcal{ALCHI}_{R^+}$. To verify this assumption, the modified tableaux algorithm will be implemented in a descendant of FaCT [Horrocks,1998].

Acknowledgements

We would like to thank the anonymous referees for their valuable comments and suggestions.

References

[Baader *et al.*, 1993] F. Baader, H.-J. Bürckert, B. Nebel, W. Nutt, and G. Smolka. On the expressivity of feature logics with negation, functional uncertainty, and sort equations. *Journal of Logic, Language and Information*, 2:1–18, 1993.

[Baader *et al.*, 1996] F. Baader, M. Buchheit, and B. Hollunder. Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1–2):195–213, 1996.

[Baader, 1990] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. Technical Report RR-90-13, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern, Germany, 1990. An abridged version appeared in *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence IJCAI-91*, pp. 446–451.

[Baader, 1991] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, 1991.

[Buchheit *et al.*, 1993] M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.

[De Giacomo & Lenzerini, 1996] G. De Giacomo and M. Lenzerini. Tbox and Abox reasoning in expressive description logics. In *Proceedings of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning (KR-96)*, pages 316–327. Morgan Kaufmann, Los Altos, 1996.

[De Giacomo & Massacci, 1998] G. De Giacomo and F. Massacci. Combining deduction and model checking into tableaux and algorithms for converse-pdl. *Information and Computation*, 1998. To appear.

[Horrocks, 1998] I. Horrocks. Using an Expressive Description Logic: FaCT or Fiction? In *Proceedings of the Sixth International Conference on the Principles of Knowledge Representation and Reasoning (KR-98)*, 1998.

[Horrocks & Gough, 1997] I. Horrocks and G. Gough. Description logics with transitive roles. In M.-C. Rousset, R. Brachmann, F. Donini, E. Franconi, I. Horrocks, and A. Levy, editors, *Proceedings of the International Workshop on Description Logics*, pages 25–28, Gif sur Yvette, France, 1997. Université Paris-Sud.

[Horrocks & Sattler, 1998] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. Technical Report 98-05, LuFg Theoretical Computer Science, RWTH Aachen, 1998. Available via www: `http://www-lti.informatik.rwth-aachen.de/Forschung/Papers.html`.

[Sattler, 1996] U. Sattler. A concept language extended with different kinds of transitive roles. In G. Görz and S. Hölldobler, editors, *20. Deutsche Jahrestagung für Künstliche Intelligenz*, volume 1137 of *Lecture Notes in Mathematics*. Springer-Verlag, 1996.

[Sattler, 1998] U. Sattler. *Terminological knowledge representation systems in a process engineering application*. PhD thesis, RWTH Aachen, 1998. To appear.

[Schild, 1991] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 466–471, Sydney, 1991.

[Schmidt-Schauß & Smolka, 1988] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with unions and complements. Technical Report SR-88-21, FB Informatik, Universität Kaiserslautern, Kaiserslautern, Germany, 1988.