Reasoning procedures

- Terminating, efficient and complete algorithms for deciding **satisfiability** – and all the other reasoning services – are available.
- Algorithms are based on tableaux-calculi techniques.
- Completeness is important for the usability of description logics in real applications.
- Such algorithms are efficient for both average and real knowledge bases, even if the problem in the corresponding logic is in PSPACE or EXPTIME.

Tableaux Calculus

The Tableaux Calculus is a decision procedure solving the problem of satisfiability.

If a formula is satisfiable, the procedure will constructively exhibit a model of the formula.

The basic idea is to incrementally build the model by looking at the formula, by decomposing it in a top/down fashion. The procedure exhaustively looks at all the possibilities, so that it can eventually prove that no model could be found for unsatisfiable formulas.

Tableaux Calculus

1. Syntactically transform a theory Σ in a *Constraint System* S – also called *tableaux*.
   Every formula of Σ is transformed into a *constraint* in S.

2. Add constraints to S, applying specific *completion rules*.
   Completion rules are either deterministic – they yield a uniquely determined constraint system – or nondeterministic – yielding several possible alternative constraint systems (*branches*).

3. Apply the completion rules until either a contradiction (a *clash*) is generated in every branch, or there is a *completed* branch where no more rule is applicable.

4. The completed constraint system gives a model of Σ; it corresponds to a particular branch of the tableaux.

The FOL example

$$\frac{\phi \wedge \psi}{\begin{array}{c}\phi\\ \psi\end{array}} \qquad \frac{\phi \vee \psi}{\phi \mid \psi} \qquad \frac{\forall x.\ \phi}{\phi\{X/t\}} \qquad \frac{\exists x.\ \phi}{\phi\{X/Z\}}$$

$$\boxed{\exists y.\ (p(y) \wedge \neg q(y)) \wedge \forall z.\, (p(z) \vee q(z))}$$

$$\exists y.\ (p(y) \wedge \neg q(y))$$

$$\forall z.\ (p(z) \vee q(z))$$

$$p(\bar{y}) \wedge \neg q(\bar{y})$$

$$p(\bar{y})$$

$$\neg q(\bar{y})$$

$$p(\bar{y}) \vee q(\bar{y})$$

$$p(\bar{y}) \qquad\qquad q(\bar{y})$$

$$< \texttt{COMPLETED} > \qquad\qquad < \texttt{CLASH} >$$

The formula is satisfiable. The devised model is $\Delta^{\mathcal{I}} = \{\bar{y}\},\ p^{\mathcal{I}} = \{\bar{y}\},\ q^{\mathcal{I}} = \emptyset$.

Negation Normal Form

Recall that the above completion rules for FOL work only if the formula has been translated into Negation Normal Form, i.e., all the negations have been pushed down.

In the same way, we can transform any $\mathcal{ALC}$ formula into an equivalent one in Negation Normal Form, so that negation appears only in front of atomic concepts:

- $\neg(C \sqcap D) \Longleftrightarrow \neg C \sqcup \neg D$
- $\neg(C \sqcup D) \Longleftrightarrow \neg C \sqcap \neg D$
- $\neg(\forall R.C) \Longleftrightarrow \exists R.\neg C$
- $\neg(\exists R.C) \Longleftrightarrow \forall R.\neg C$

Completion Rules: the AND rule

The propagation rules come straightforwardly from the semantics of constructors.

If in a given interpretation $\mathcal{I}$, whose domain contains the element a, we have that $a \in (C \sqcap D)^{\mathcal{I}}$, then from the semantics we know that such element a should be in the intersection of $C^{\mathcal{I}}$ and $D^{\mathcal{I}}$, i.e. it should be in both $C^{\mathcal{I}}$ and $D^{\mathcal{I}}$.

Since this must be true for any interpretation, we can abstract from interpretations and their elements, and say that if in a generic interpretation we have a generic element x that is in the interpretation of the concept $C \sqcap D$ (denote this by $x : (C \sqcap D)$) then the element x should belong both to the interpretation of C and to the interpretation of D.

The AND rule

Suppose now we want to construct a generic interpretation S such that the set corresponding to the concept $C \sqcap D$ contains at least one element. We can state this initial requirement as the constraint $x : (C \sqcap D)$.

Following the semantics, we know that S must be such that the constraints $x : C$ and $x : D$ must hold, hence we can add these new constraints to S, knowing that if S will ever satisfy them then it will also satisfy the first constraint.

These considerations lead to the following propagation rule:

$$S \rightarrow_{\sqcap} \{x : C,\ x : D\} \cup S$$

if 1. $x : C \sqcap D$ is in S,

2. $x : C$ and $x : D$ are not both in S

The SOME rule

If in a given interpretation $\mathcal{I}$, whose domain contains the element a, we have that $a \in (\exists R.C)^{\mathcal{I}}$, then from the semantics we know that there must be an element b (not necessarily distinct from a) such that $(a, b) \in R^{\mathcal{I}}$, and $b \in C^{\mathcal{I}}$.

Since this must be true for any interpretation, we can abstract from interpretations and their elements, and say that if in a generic interpretation we have a generic element x that is in the interpretation of the concept $\exists R.C$ (denote this by $x : \exists R.C$) then there must be a generic element y such that x and y are in relation through R (denote it xRy) and y belongs to the interpretation of C (denoted as $y : C$).

The SOME rule *(cont.)*

These considerations lead to the following propagation rule:

$$S \;\rightarrow_{\exists}\; \{xRy,\; y : C\} \cup S$$

if 1. $x : \exists R.C$ is in S,

2. y is a new variable,

3. there is no z such that

   both xRz and $z : C$ are in S

Completion rules for $\mathcal{ALC}$

$S \rightarrow_{\sqcap} \{x : C,\ x : D\} \cup S$

if 1. $x : C \sqcap D$ is in S,

2. $x : C, x : D$ are not both in S

$S \rightarrow_{\sqcup} \{x : E\} \cup S$

if 1. $x : C \sqcup D$ is in S,

2. neither $x : C$ nor $x : D$ is in S,

3. $E = C$ or $E = D$

$S \rightarrow_{\forall} \{y : C\} \cup S$

if 1. $x : \forall R.C$ is in S,

2. xRy is in S,

3. $y : C$ is not in S

$S \rightarrow_{\exists} \{xRy,\ y : C\} \cup S$

if 1. $x : \exists R.C$ is in S,

2. y is a new variable,

3. there is no z such that

both xRz and $z : C$ are in S

Clash

While building a constraint system, we can look for evident contradictions to see if the constraint system is not satisfiable. We call these contradictions **clashes**.

A *clash* is a constraint system having the form:

$\{x : A,\ x : \neg A\}$, where A is a concept name.

A clash is evidently an unsatisfiable constraint system, hence any constraint system containing a clash is obviously unsatisfiable.

An Example of tableaux

Satisfiability of the concept:

$$\boxed{\big((\forall \texttt{CHILD}.\texttt{Male}) \sqcap (\exists \texttt{CHILD}.\neg\texttt{Male})\big)}$$

$$\big((\forall \texttt{CHILD}.\texttt{Male}) \sqcap (\exists \texttt{CHILD}.\neg\texttt{Male})\big)(\texttt{x})$$

$$(\forall \texttt{CHILD}.\texttt{Male})(\texttt{x}) \qquad \sqcap\textit{-rule}$$

$$(\exists \texttt{CHILD}.\neg\texttt{Male})(\texttt{x}) \qquad \text{“}$$

$$\texttt{CHILD}(x, y) \qquad \exists\textit{-rule}$$

$$\neg\texttt{Male}(\boldsymbol{y}) \qquad \text{“}$$

$$\texttt{Male}(\boldsymbol{y}) \qquad \forall\textit{-rule}$$

$$\langle CLASH \rangle$$

An Example of tableaux - *constraint* syntax -

$\boxed{\big((\forall \texttt{CHILD}.\texttt{Male}) \sqcap (\exists \texttt{CHILD}.\neg\texttt{Male})\big)}$

$x : \big((\forall \texttt{CHILD}.\texttt{Male}) \sqcap (\exists \texttt{CHILD}.\neg\texttt{Male})\big)$	
$x : (\forall \texttt{CHILD}.\texttt{Male})$	$\sqcap$-*rule*
$x : (\exists \texttt{CHILD}.\neg\texttt{Male})$	"
$x\ \texttt{CHILD}\ y$	$\exists$-*rule*
$y : \neg\texttt{Male}$	"
$y : \texttt{Male}$	$\forall$-*rule*
$\langle CLASH \rangle$	

Another example

$$\boxed{((\forall \texttt{CHILD.Male}) \sqcap (\exists \texttt{CHILD.Male}))}$$

$x : ((\forall \texttt{CHILD.Male}) \sqcap (\exists \texttt{CHILD.Male}))$

$x : (\forall \texttt{CHILD.Male})$	$\sqcap$-*rule*
$x : (\exists \texttt{CHILD.Male})$	"
$x \ \texttt{CHILD} \ y$	$\exists$-*rule*
$y : \texttt{Male}$	"
$y : \texttt{Male}$	$\forall$-*rule*

$\langle COMPLETED \rangle$

Exercise: find a model.

Tableaux with individuals

Check the satisfiability of the ABox:

$(\texttt{Parent} \sqcap \forall \texttt{CHILD}.\texttt{Male})(\texttt{john})$

$\neg \texttt{Male}(\texttt{mary})$

$\texttt{CHILD}(\texttt{john}, \texttt{mary})$

$\texttt{john} : \texttt{Parent} \sqcap \forall \texttt{CHILD}.\texttt{Male}$

$\texttt{mary} : \neg \texttt{Male}$

$\texttt{john}\ \texttt{CHILD}\ \texttt{mary}$

$\texttt{john} : \texttt{Parent}$ $\sqcap$-*rule*

$\texttt{john} : \forall \texttt{CHILD}.\texttt{Male}$ "

$\texttt{mary} : \texttt{Male}$ $\forall$-*rule*

$\langle CLASH \rangle$

The knowledge base is inconsistent.

Soundness of the Tableaux for $\mathcal{ALC}$

The calculus does not add unnecessary contradictions.

That is, deterministic rules always preserve the Satisfiability of a constraint system, and nondeterministic rules have always a choice of application that preserves Satisfiability.

Termination of the Tableaux for $\mathcal{ALC}$

A constraint system is *complete* if no propagation rule applies to it. A complete system derived from a system S is also called a *completion* of S. Completions are reached when there is no infinite chain of applications of rules.

Intuitively, this can be proved by using the following argument: all rules but $\longrightarrow_\forall$ are never applied twice on the same constraint; this rule in turn is never applied to a variable x more times than the number of the *direct successors* of x, which is bounded by the length of a concept; finally, each rule application to a constraint $y : C$ adds constraints $z : D$ such that D is a strict subexpression of C.

Completeness of the Tableaux for $\mathcal{ALC}$

If S is a completion of $\{x : C\}$ and S contains no clash, then it is always possible to construct an interpretation for C on the basis of S, such that $C^{\mathcal{I}}$ is nonempty.

The proof is a straightforward induction on the length of the concepts involved in each constraint.