Part I

Logic Engineering

ocurre en todos los casos
en que participa la verdad
que se transforma todo sentido aparente
espejo falso
en lugares ciertos entre la nada y el infinito

from "Guitarra Negra," Luis Alberto Spinetta

For many years "Logic" was "Classical Logic," mainly classical first-order logic, and there were good reasons for this. To mention some, first-order logic offers high expressive power, simplicity, good behavior (both syntactically and semantically), and a clean and well developed model theory.

This is just a complicated way of saying that first-order logic is beautiful...for many tasks. But when we think about applications requiring *effective inference*, first-order logic is simply *not* the choice: its satisfiability problem — i.e., the problem of determining whether there exists a model in which a given first-order formula is true — is not decidable. In addition, first-order logic sometimes does not measure up to the task at hand. It cannot, for example, capture the fact that one relation is the transitive closure of another one, and this might be crucial for a certain modeling task.

For these reasons mainly, first-order logic has been loosing its privileged position as a representation formalism in many areas where applications requiring effective inference methods are central, such as Artificial Intelligence, Knowledge Representation, Computational Linguistics, Software Design and Verification, or Databases. In these fields, the applications themselves have given rise to new formalisms, specially tailored for the problems to be addressed. In some cases, like in the early days of Artificial Intelligence, this growth has even been chaotic, with hundreds of new proposals, and very restricted means to evaluate them.

As an answer to this problem, a new field of Logic Engineering is starting to develop. To judge the appropriateness of the name, consider the definition of *engineer* [Davidson *et al.*, 1994]

> **engineer:** *n.* one who designs or makes, or puts to practical use [...]

In line with its name, Logic Engineering studies ways to construct new formalisms, with good properties like decidability, appropriate expressive power, effective reasoning methods, and good meta-logical characteristics (completeness, interpolation, etc.), for a given, particular need.

How do we design "made-to-fit" logics? That is the topic we will discuss in this part of the thesis.

Chapter 1

Cutting Out Fragments

The logic is invariant,...
but the data are different.
So the results are different.

from "Stranger in a Strange Land," Robert Heinlein

1.1 Looking for the Right Language

The rules of the game are set: we are searching for good formal languages for specific tasks, where by "formal languages" we mean *languages with a precise syntax and semantics*, and by "specific tasks" we mean *reasoning tasks*. In particular, we will focus on reasoning tasks involving *inference* and hence a further condition is required on our notion of formal language, namely it should provide a calculus defining some kind of consequence relation.

We might as well start by looking into why first-order logic (FO) is not necessarily the best choice in all cases. First, it might be too complex for the reasoning task we have to address.

THEOREM 1.1. *The satisfiability problem for first-order logic is not decidable.*

If we are in search of effective inference mechanisms, Theorem 1.1 immediately disqualifies first-order logic. Of course, if what we require are languages for specifying properties to *verify* in a given model, i.e. if our problem requires only model checking, then FO enters the list of candidates again. But on the other hand, even with all its expressive power, it is possible that FO just does not provide exactly the needed expressivity. For example,

THEOREM 1.2. *First-order logic cannot define the transitive closure of a relation.*

Theorems 1.1 and 1.2 show that there might be different reasons why we need to look for alternatives to FO, in addition they will also let us introduce a number of definitions and notions that will keep coming up as a central theme in the thesis, like the importance of formal syntax and semantics, the use of model-theoretical techniques like reduction arguments and games, comparisons of expressive power, etc. To make things crystal clear, let us start by formally introducing first-order logic.

DEFINITION 1.3. [First-order language] Let $\mathsf{REL} = \{R_1, R_2, \ldots\}$ be a countable set of *relation symbols*, $\mathsf{FUN} = \{f_1, f_2, \ldots\}$ a countable set of *function symbols*, $\mathsf{CON} =$

$\{c_1, c_2, \ldots\}$ a countable set of *constant symbols* and $\mathsf{VAR} = \{x_1, x_2, \ldots\}$ a countable set of *variables*. We assume that REL, FUN, CON and VAR are pairwise disjoint. To each relation symbol $R_i \in \mathsf{REL}$ and each function symbol $f_i \in \mathsf{FUN}$ we associate an arity $n > 0$. We call $\mathcal{S} = \langle \mathsf{REL}, \mathsf{FUN}, \mathsf{CON}, \mathsf{VAR}\rangle$ a *signature*, and we will sometimes focus on *relational signatures* where $\mathsf{FUN} = \{\}$. This is usually not a restriction as we can represent functions as constrained relations.

The *well-formed terms* of the first-order language over the signature $\langle \mathsf{REL}, \mathsf{FUN}, \mathsf{CON}, \mathsf{VAR}\rangle$ are

$$\mathsf{TERMS} := x_i \mid c_i \mid f_i(t_1, \ldots, t_n),$$

where, $x_i \in \mathsf{VAR}$, $c_i \in \mathsf{CON}$, $f_i \in \mathsf{FUN}$ of arity n and $t_1, \ldots, t_n \in \mathsf{TERMS}$. The *well-formed formulas* over the signature are

$$\mathsf{FORMS} := \top \mid t_1 = t_2 \mid R_i(t_1, \ldots, t_n) \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \exists x_i.\varphi,$$

where $t_1, t_2, \ldots, t_n \in \mathsf{TERMS}$, $R_i \in \mathsf{REL}$ is an n-ary relation symbol, $\varphi, \varphi_1, \varphi_2 \in \mathsf{FORMS}$ and $x_i \in \mathsf{VAR}$. As usual, we take $\vee, \rightarrow, \leftrightarrow$ and $\forall$ as defined symbols.

Turning to semantics, first-order formulas are interpreted on first-order models.

Definition 1.4. [First-order models and satisfiability] A first-order *model* for a signature $\mathcal{S}$, is a structure $\mathcal{M} = \langle M, \cdot^{\mathcal{M}}\rangle$ where M is a non-empty set and $\cdot^{\mathcal{M}}$ is an *interpretation function* defined over $\mathsf{REL} \cup \mathsf{FUN} \cup \mathsf{CONS}$ such that $\cdot^{\mathcal{M}}$ assigns an n-ary relation over M to n-ary relation symbols in REL, an n-ary function ${}^nM \rightarrow M$ to n-ary function symbols in FUN, and an element in M to constant symbols in CONS. When the signature is small, we will simple write $\mathcal{M} = \langle M, \{R_i^{\mathcal{M}}\}, \{f_j^{\mathcal{M}}\}, \{c_k^{\mathcal{M}}\}\rangle$ instead of $\mathcal{M} = \langle M, \cdot^{\mathcal{M}}\rangle$.

An *assignment* g for $\mathcal{M}$ is a mapping $g : \mathsf{VAR} \rightarrow M$. Given an assignment g for $\mathcal{M}$, $x \in \mathsf{VAR}$ and $m \in M$, we define g^x_m (an *x-variant* of g) by $g^x_m(x) = m$ and $g^x_m(y) = g(y)$ for $x \neq y$. Given a model $\mathcal{M}$ and an assignment g for $\mathcal{M}$, the interpretation function $\cdot^{\mathcal{M}}$ can be extended to all elements in TERMS:

$$\begin{array}{rcl} x_i^{\mathcal{M}} & = & g(x_i) \\ f(t_1, \ldots, t_n)^{\mathcal{M}} & = & f^{\mathcal{M}}(t_1^{\mathcal{M}}, \ldots, t_n^{\mathcal{M}}). \end{array}$$

Finally the *satisfiability relation* $\models$ is defined as

$$\begin{array}{rll} \mathcal{M} \models \top[g] & & \text{always} \\ \mathcal{M} \models t_1 = t_2[g] & \text{iff} & t_1^{\mathcal{M}} = t_2^{\mathcal{M}} \\ \mathcal{M} \models R(t_1, \ldots, t_n)[g] & \text{iff} & R^{\mathcal{M}}(t_1^{\mathcal{M}}, \ldots, t_n^{\mathcal{M}}) \\ \mathcal{M} \models \neg\varphi[g] & \text{iff} & \mathcal{M} \not\models \varphi[g] \\ \mathcal{M} \models \varphi_1 \wedge \varphi_2[g] & \text{iff} & \mathcal{M} \models \varphi_1[g] \text{ and } \mathcal{M} \models \varphi_2[g] \\ \mathcal{M} \models \exists x_i.\varphi[g] & \text{iff} & \mathcal{M} \models \varphi[g^{x_i}_m] \text{ for some } m \in M. \end{array}$$

If a given formula φ is satisfied under every assignment for $\mathcal{M}$, we say that φ is *valid* in $\mathcal{M}$ and write $\mathcal{M} \models \varphi$. For a sentence φ, let $Mod(\varphi) = \{\mathcal{M} \mid \mathcal{M} \models \varphi\}$.

A piece of notation now, for a given language $\mathcal{L}$, let $Sat(\mathcal{L})$ denote the satisfiability problem for $\mathcal{L}$ (i.e., the problem of deciding wether a formula of $\mathcal{L}$ is satisfiable), and $Val(\mathcal{L})$ the dual validity problem.

We are now ready for the proof of Theorem 1.1. In 1936, Church put forward a bold thesis: every computable function from natural numbers to natural numbers is recursive in the sense of Herbrand-Gödel-Kleene. He then showed that no recursive function could decide the validity of first-order sentences, and concluded that there was no decision algorithm for $Sat(\mathsf{FO})$ [Church, 1936]. Independently, a different proof was published by Turing a year later [Turing, 1937], by formalizing Turing machines by means of first-order formulas, and reducing an undecidable class of particular word problems for Turing machines to the validity problem of the encoding. But a very simple proof can be given by means of an undecidable tiling or domino problem [Berger, 1966].

Definition 1.5. [The $\mathbb{N} \times \mathbb{N}$ tiling problem] A *tiling system* is a triple $\mathcal{T} = \langle T, H, V\rangle$ where T is a finite set (the set of tiles), and $H, V \subseteq T \times T$ are relations expressing horizontal and vertical compatibility constraints between the tiles.

We say that $\mathcal{T}$ *tiles* $\mathbb{N} \times \mathbb{N}$ if there exists a *tiling function* $t : \mathbb{N} \times \mathbb{N} \to T$ such that for all $(x, y) \in \mathbb{N} \times \mathbb{N}$, whenever $t(x, y) = t_1$, $t(x + 1, y) = t_2$ and $t(x, y + 1) = t_3$ then $H(t_1, t_2)$ and $V(t_1, t_3)$.

In [Berger, 1966] the undecidability of the $\mathbb{N} \times \mathbb{N}$ tiling problem is proved by reducing the halting problem for Turing machines to the problem of deciding whether a tiling function exists, for any given tiling system. Tiling problems are also very useful as a means to establish complexity lower bounds [Chlebus, 1986].

Proof of Theorem 1.1. Consider the following encoding into FO of a given tiling system $\mathcal{T} = \langle\{t_1, \ldots, t_n\}, H, V\rangle$ (from [Gurevich, 1976]). For $h, v, t_1, \ldots, t_n$ unary functions, let $\varphi(x)$ be the conjunction of the following formulas

$$
\begin{gathered}
h(v(x)) = v(h(x)), \\
\bigvee\{t_i(x) = x \mid 1 \leq i \leq n\}, \\
\bigwedge\{t_i(x) = x \to \neg(t_j(x) = x) \mid 1 \leq i < j \leq n\}, \\
\bigvee\{t_i(x) = x \wedge t_j(h(x)) = h(x) \mid H(t_i, t_j)\}, \\
\bigvee\{t_i(x) = x \wedge t_j(v(x)) = v(x) \mid V(t_i, t_j)\}.
\end{gathered}
$$

It is almost trivial to verify that $\forall x.\varphi(x)$ is satisfiable iff $\mathcal{T}$ tiles $\mathbb{N} \times \mathbb{N} \to T$. From right to left, let t be a tiling function, then $\mathcal{M} = \langle \mathbb{N} \times \mathbb{N}, \cdot^{\mathcal{M}}\rangle$, where

$$
\begin{array}{rcll}
t_i^{\mathcal{M}}(a, b) & = & (a, b) & \text{if } \ t(a, b) = t_i \\
t_i^{\mathcal{M}}(a, b) & = & (a + 1, b + 1) & \text{if } \ t(a, b) \neq t_i \\
h^{\mathcal{M}}(a, b) & = & (a + 1, b) & \\
v^{\mathcal{M}}(a, b) & = & (a, b + 1) &
\end{array}
$$

satisfies $\forall x.\varphi(x)$. For the other direction, assume $\mathcal{M} \models \forall x.\varphi(x)$ and let $a \in M$ be fixed but arbitrary, then the function

$$t(n, m) = t_i \quad \text{iff} \quad \mathcal{M} \models t_i(h^n(v^m(x))) = h^n(v^m(x))[a]$$

is a tiling function. QED

As we see from the proof, very little of first-order expressive power is needed to arrive at undecidability. If the language has a countable number of unary function symbols and equality then only one variable is needed, and the formulas are purely universal. As we already said, we can always trade the function symbols used in the encoding for relational symbols, and use the language itself to force them to behave as functions (enforcing totality $\forall x.\exists y.(R(x,y))$ and functionality $\forall x y_1 y_2.((R(x,y_1) \wedge R(x,y_2)) \rightarrow y_1 = y_2))$, but notice that we pay a price on more variables and a more complicated pattern of quantification.

Studying this kind of trade-off is a long standing project of classical logic, and attempts to map decidable and undecidable fragments of FO are varied [Ackermann, 1954; Lewis, 1979; Börger *et al.*, 1997]. We will elaborate on this point in Section 1.2, when we analyze different fragments of FO.

The discussion so far may be summarized by saying that FO is, in some cases, *too expressive*: it allows us to encode undecidable problems. At the same time it is often not expressive enough. A typical example when expressivity beyond first-order logic is called for, is the need to refer to the transitive closure of a relation in a model. As we stated in Theorem 1.2, first-order logic lacks the expressive power required to define the transitive closure of a relation. For the proof of the theorem we will introduce another important tool: a way to compare models.

Suppose we want to prove that a certain property P is not expressible in first-order logic. One way of showing this is to exhibit two first-order models, which agree on all first-order sentences, but such that P is true in one and false in the other. This argument can be refined further: suppose we are able to prove that for any $k \in \mathbb{N}$ there are two first-order models $\mathcal{M}_k$ and $\mathcal{N}_k$ such that they agree *on all first-order sentences up to quantifier rank* k but they disagree on the truth value of P. This is enough, as any first-order sentences defining P will have a certain fixed finite quantifier rank.

Ehrenfeucht [1961] established a strong correlation between the truth of sentences of quantifier rank k in a pair of models, and the rounds of a model comparison game. Let us define things formally. The notion of partial isomorphism is one of the crucial ingredients.

Definition 1.6. [Partial isomorphism] Let $\mathcal{M}$ and $\mathcal{N}$ be first-order models over a relational signature $\mathcal{S}$. Let $p : M \rightarrow N$ be a partial map. p is said to be a *partial isomorphism* from $\mathcal{M}$ to $\mathcal{N}$ iff

i. p is injective,
ii. $\forall c \in \mathsf{CONS}$, $c^{\mathcal{M}} \in dom(p)$ and $p(c^{\mathcal{M}}) = c^{\mathcal{N}}$,
iii. $\forall R \in \mathsf{REL}$, and $a_1, \ldots, a_n \in dom(p)$, $R^{\mathcal{M}}(a_1, \ldots, a_n)$ iff $R^{\mathcal{N}}(p(a_1), \ldots, p(a_n))$.

Notice that, in general, partial isomorphisms do not preserve the validity of formulas with quantifiers as isomorphisms do. But interestingly, they allow us to introduce a notion of *extensions* which will match the restriction to a fixed quantifier rank. This will become clear in Definition 1.8 below. First, we introduce the game-theoretical perspective, which is more intuitive in many cases.

Definition 1.7. [Ehrenfeucht games] Let $\mathcal{M}$ and $\mathcal{N}$ be first-order models over the same relational signature. The *Ehrenfeucht game* $G_k(\mathcal{M}, \mathcal{N})$ is played by two players called the *Spoiler* and the *Duplicator*. Each player has to make k moves in the course of a play. The players take turns. In his i-th move the Spoiler first selects a structure, $\mathcal{M}$ or $\mathcal{N}$, and an element in this structure. If the Spoiler chooses e_i in $\mathcal{M}$, then the Duplicator in his i-th move chooses an element f_i in $\mathcal{N}$. If the Spoiler chooses f_i in $\mathcal{N}$, then the Duplicator chooses an element e_i in $\mathcal{M}$. The Duplicator *wins* the game if $\bar{e} \mapsto \bar{f}$ is a partial isomorphism from $\mathcal{M}$ to $\mathcal{N}$. Otherwise the Spoiler wins. We say that one of the players has a *winning strategy* in $G_k(\mathcal{M}, \mathcal{N})$, if it is possible for him to win each play whatever choices are made by the opponent.

Ehrenfeucht games provide a natural characterization of elementary equivalence up to rank k. If Duplicator has a winning strategy for the game $G_k(\mathcal{M}, \mathcal{N})$, then $\mathcal{M}$ and $\mathcal{N}$ agree on all sentences up to quantifier rank k (notation $\mathcal{M} \equiv_k \mathcal{N}$). The advantage of Ehrenfeucht games is that the existence of a winning strategy for Duplicator is often *easy to grasp*. Their disadvantage is that sometimes, arguments proving this are *hard to describe*.

Instead, Fraïssé [1954] provided a purely algebraic characterization by means of k-back-and-forth systems.

Definition 1.8. [k-back-and-forth systems] Let $\mathcal{M}$ and $\mathcal{N}$ be two first-order models on a relational signature. A *k-back-and-forth system* is a sequence $(I_j)_{j \leq k}$ with the following properties:

i. Every I_j is a non-empty set of partial isomorphisms from $\mathcal{M}$ to $\mathcal{N}$.
ii. (Forth property) $\forall j < k$, $p \in I_{j+1}$, and $m \in M$ there is $q \in I_j$ extending p and $m \in dom(q)$.
iii. (Back property) $\forall j < k$, $p \in I_{j+1}$, and $n \in N$ there is $q \in I_j$ extending p and $n \in ran(q)$.

We say that $\mathcal{M}$ and $\mathcal{N}$ are *k-isomorphic* (notation $\mathcal{M} \cong_k \mathcal{N}$) if there is a k-back-and-forth system between them.

The pieces are put together in the following result.

Theorem 1.9. [Ehrenfeucht-Fraïssé Theorem] *Given two first-order models $\mathcal{M}$ and $\mathcal{N}$ on a relational signature, and $k \in \mathbb{N}$, the following are equivalent*

i. Duplicator has a winning strategy in $G_k(\mathcal{M}, \mathcal{N})$.
ii. $\mathcal{M} \cong_k \mathcal{N}$.
iii. $\mathcal{M} \equiv_k \mathcal{N}$.

We have now all the machinery we need to embark on the proof of Theorem 1.2 (from [Ebbinghaus and Flum, 1999]).

Proof of Theorem 1.2. We want to show that the relation $TC(R) = R^+$ of transitive closure is not first-order definable. We start by proving a simpler claim. A first-order model over the relational signature $\langle \{R\}, \{\}, \{\}, \{\} \rangle$, where R is binary is called a (simple, possibly infinite) *graph*. We say that a class C of models in this signature is first-order definable if there exists a first-order sentence φ such that $\mathsf{C} = Mod(\varphi)$. We say that a graph $\langle M, R \rangle$ is *connected* if any two different elements in M are related in R^+, the transitive closure of R.

Claim 1.10. The class CONN of connected graphs is not first-order definable.

Proof of Claim. For $l \in \mathbb{N}$, let $\mathcal{C}_l$ be the graph given by a cycle of length $l+1$:

$$\mathcal{C}_l = \langle \{0, \ldots, l\}, \{(i, i+1), (i+1, i) \mid 0 \leq i < l\} \cup \{(0, l), (l, 0)\} \rangle.$$

We will show that

$$\text{for every } k \in \mathbb{N}, l \geq 2^k,\ \mathcal{C}_l \equiv_k \mathcal{C}_l \uplus \mathcal{C}_l, \tag{1.1}$$

where $\uplus$ denotes disjoint union. The claim follows from (1.1), because suppose φ defines CONN, and let k be the quantifier rank of φ. As $\mathcal{C}_{2^k}$ is connected, then $\mathcal{C}_{2^k} \models \varphi$, as $\mathcal{C}_{2^k} \equiv_k \mathcal{C}_{2^k} \uplus \mathcal{C}_{2^k}$ then also $\mathcal{C}_{2^k} \uplus \mathcal{C}_{2^k} \models \varphi$, and $\mathcal{C}_{2^k} \uplus \mathcal{C}_{2^k} \in Mod(\varphi) = \mathsf{CONN}$, a contradiction.

We use Theorem 1.9 to prove (1.1). Intuitively, the winning strategy of Duplicator works by picking points which are sufficiently far apart. Whenever Spoiler tries to signal a difference between $\mathcal{C}_l$ and $\mathcal{C}_l \uplus \mathcal{C}_l$ by choosing points in the different connected components of $\mathcal{C}_l \uplus \mathcal{C}_l$, during round j, Duplicator chooses points in $\mathcal{C}_l$ which are at least at a distance 2^j. This is enough for the logic to think of them as unreachable. Formally, for a graph $\mathcal{G}$, let $d^{\mathcal{G}} : G \times G \to \mathbb{N}$ be the length of a minimal path between two elements of $\mathcal{G}$ if such path exists, or ∞ otherwise. Define for $j \in \mathbb{N}$, the "truncated" j-distance function $d_j^{\mathcal{G}}$ by

$$d_j^{\mathcal{G}}(e, e') = \begin{cases} d^{\mathcal{G}}(e, e') & \text{if } d^{\mathcal{G}}(e, e') < 2^{j+1}, \\ \infty & \text{otherwise.} \end{cases}$$

It is easy to verify that $(I_j)_{j \leq m}$ is an m-back-and-forth system between $\mathcal{C}_l$ and $\mathcal{C}_l \uplus \mathcal{C}_l$ if $l \geq 2^m$, where $I_j = \{p \mid d_j^{\mathcal{C}_l}(e, e') = d_j^{\mathcal{C}_l \uplus \mathcal{C}_l}(p(e), p(e')),\ \text{for } e, e' \in dom(p)\}$. $\dashv$

From the claim, it follows that $TC(R)$ is not first-order definable. For suppose it were definable; let $\varphi(x, y)$ be a first-order formula defining $TC(R)$, i.e., $\mathcal{M} \models \varphi[m, m']$ iff $(m, m') \in R^+$. Then CONN can be defined as the class of models $\{\mathcal{M} \mid \mathcal{M} \models \forall xy.(\neg(x = y) \to \varphi(x, y))\}$. QED

After all the work we have done in this section we can happily agree that first-order logic comes with many powerful techniques, which is perhaps one of its strongest points. But these same tools let us establish results like Theorems 1.1 and 1.2, which signal important weaknesses when we have certain kind of applications in mind.

In the chapters to come we will discuss how fragments of FO, or fragments of extensions of FO, can be fine-tuned to provide effective reasoning methods, good meta-logical properties, better complexity results, and the exact expressive power needed.

1.2 Fragments of FO, and Extensions

But, there are continuum many different fragments. Obviously, we don't want to consider all of them. How do we define "nice fragments"? A first possibility is to restrict attention to classes presentable in some fixed way. For example, to consider finite classes (i.e., classes with a finite number of models) would be an option, but these classes might be of little applicability if we are interested in inference.

General syntactic and semantic constraints are probably better suited for the job. For example, we could use the semantic consequence relation: for a sentence α, let $Cn(\alpha)$ be the set of all first-order consequences of α. We can now consider the following problem, given a sentence α, decide whether the decision problem for $Cn(\alpha)$ is decidable (i.e., whether for an arbitrary sentence β it is possible to decide if $\beta \in Cn(\alpha)$). A simple reduction argument shows that this problem is as hard as $Sat(\mathsf{FO})$ itself. Another possibility is to choose a certain class of structures (lattices, rings, ...) and attempt to characterize its first-order theory, but this seems to be more the work of (perhaps Universal) Algebra.

In Logic, the syntactic path has been (much) more prevalent. For a long time, logicians have been interested in classes of formulas defined by simple syntactic restrictions. In [1915], Löwenheim gave a decision procedure for the satisfiability of sentences with only unary predicates. He also proved that sentences with only binary predicates form a reduction class for validity, i.e., this set of sentences is such that there is a recursive function f from first-order sentences into sentences with only binary predicates with the property that φ is valid iff $f(\varphi)$ is. Löwenheim's undecidability result was sharpened by Kalmár in [1936]: one binary predicate suffices. This implies, for example, that the first-order theory of graphs is undecidable.

Even if we restrict ourselves to syntactic ways of cutting out fragments, there are different possibilities.

1.2.1 Prenex Normal Form Fragments

Some of the most familiar fragments of first-order logic are defined by means of restrictions on the quantifier prefix of formulas in prenex normal form. As is standard, we will refer to prenex normal form fragments by using finite strings over $\{\exists, \exists^*, \forall, \forall^*\}$. For example, the string $\forall\exists^*$ represents the class of first-order formulas in prenex normal form, where the quantifier prefix starts with a universal quantifier which is followed by zero or more existential quantifiers.

Theorem 1.1 shows that if functional symbols are allowed in the language, then the simple $\forall$ fragment is already undecidable. As is traditional, in what follows we will only discuss fragments over relational signatures. Here are some of the best known results. In 1920, Skolem showed that $\forall^*\exists^*$ sentences form a reduction class for satisfiability. In 1928, Bernays and Schönfinkel gave a decision procedure for the satisfiability of $\exists^*\forall\exists^*$ sentences. Gödel, Kalmár and Schütte, independently in 1931, 1933 and 1934 respectively, discovered decision procedures for the satisfiability of $\exists^*\forall^2\exists^*$ sentences. In 1933, Gödel showed that $\forall^3\exists^*$ sentences form a reduction class for satisfiability. More recently, Kahr in 1962 proved the undecidability of $Sat(\forall\exists\forall)$.

These results are interesting and useful, as is witnessed by the following immediate application. Suppose we built a formal specification (for example, of a certain mathematical structure) by means of first-order sentences which enumerate the needed properties. If all the formulas used lie inside one of the decidable classes mentioned above, then there exist a completely mechanic way of determining if such a structure exists (i.e., by checking the consistency of the specification).

The collection of similar results was eventually organized in what is known today as the *Classification Problem* for first-order logic:

- Which fragments are decidable for satisfiability and which are undecidable?
- Which fragments are decidable for finite satisfiability (satisfiable in a finite model) and which are undecidable?
- Which fragments have the finite model property and which contain axioms of infinity (that is, satisfiable formulas without finite models)?

Finite models are important because for any fragment F of first-order logic having the finite model property, $Sat(\mathsf{F})$ can be decided provided that we can decide membership of a formula in F. The classification problem for prenex normal form fragments admits a complete finite solution. This follows from the Classifiability Theorem of Gurevich [1969]. Also the complexity, for the cases where the fragment is decidable, is fairly well mapped out (see [Börger *et al.*, 1997] for the most up-to-date account).

In other words, the issue of decidability/undecidability for prenex normal form fragments of FO is pretty much settled. But in other aspects these fragments are not satisfactory at all. In almost all cases, the study of their meta-logical properties is difficult as they don't have a neat model theory. The restrictions are "too syntactical" in nature for issues like axiomatization or semantic characterization to be manageable.

1.2.2 Finite Variable Fragments

Let FO^k be the restriction of first-order logic over a relational signature to formulas that contain only the variables $x_1, \ldots, x_k$, from some fixed enumeration of VAR. Finite-variable fragments of first-order logic were introduced for technical reasons in [Henkin, 1967]. Logics with only a finite number of variables are important in many branches of mathematical logic and its applications, including finite model theory, model checking, database query languages and knowledge representation. Notice that these fragments cannot be "turned into prenex normal form." In the transformation to prenex normal form new variables are needed to push out quantifiers. In fact, the clever reuse of variables is crucial.

Example 1.11. [Börger *et al.*, 1997] Given a graph $\mathcal{G}$, we need $n+1$ variables to characterize a path of length n in $\mathcal{G}$, if we require the formula to be in prenex normal form

$$\exists x_0 \ldots x_n.(\bigwedge_{0 \leq i < n} E(x_i, x_{i+1})),$$

but a formula in FO^2 is enough if we allow reuse

$$\exists x_0 x_1.(E(x_0, x_1) \wedge \exists x_0.(E(x_1, x_0) \wedge \ldots)).$$

The satisfiability problem for FO^k is undecidable, even for formulas without equality, for all $k \geq 3$, given that FO^3 already extends the prefix class $\forall\exists\forall$. That FO^2 (without equality) is decidable was first proved in [Scott, 1962], the result was extended to the language with equality in [Mortimer, 1975].

Theorem 1.12. FO^2 *has the finite model property, hence* $Sat(\mathsf{FO}^2)$ *is decidable.*

In [Grädel *et al.*, 1997a] the complexity upper bound set by Mortimer is improved to an essentially optimal single exponential in the model size, proving $Sat(\mathsf{FO}^2) \in \mathrm{NT{\scriptsize IME}}(2^{O(n)})$. This result can be extended to FO^2 with constant symbols, but that seems to be the decidability boundary: the *Sat* problem of FO^2 (even without equality) extended with a single unary function is undecidable. Also, the satisfiability problem for the one variable fragment of FO is undecidable if unary functions are allowed.

[Grädel *et al.*, 1997b; Pacholski *et al.*, 1997; Grädel *et al.*, 1999] provide a detailed analysis of (un)decidability phenomena for FO^2 and extensions. A particularly interesting case (see Section 2.2) is C^2, the extension of FO^2 (with equality) by counting quantifiers $\exists^{\geq m}$ and $\exists^{\leq m}$ for $m \in \mathbb{N}$. It is almost immediate to see that C^2 does not have the finite model property. The formula

$$\forall x.\exists^{=1} y.R(x,y) \wedge \forall y.\exists^{\leq 1} x.R(x,y) \wedge \exists y.\forall x.\neg R(x,y)$$

forces R to be the graph of an injective and not surjective function in the domain, which can only be satisfied on an infinite model. However, $Sat(\mathsf{C}^2)$ is shown to be NEXPTIME-complete if unary coding for numbers in quantifiers is used.

For further results and critical discussions of pros and cons of FO^k, see [Andréka *et al.*, 1998]. This paper shows failure of the Łos-Tarski and Craig interpolation theorems inside finite variable fragments. In [Areces and Marx, 1998], failure of (even weak) interpolation in FO^k for $k \geq 2$ is proved as a corollary of a more general result concerning failure of the interpolation property. A short direct proof is also provided, showing that very little is needed to achieve failure of interpolation in finite variable fragments. Let $\vdash_2$ be the derivation system defined as follows

- Ax1 Every propositional tautology is an axiom scheme.
- Ax2$_i$ $\forall x_i.(\varphi \to \psi) \to (\forall x_i.\varphi \to \forall x_i.\psi)$, for $i \in \{0,1\}$.
- Ax3 $\forall x_1 x_0.\varphi \to \forall x_0 x_1.\varphi$.
- MP From φ and $\varphi \to \psi$ infer ψ.
- UG$_i$ From φ infer $\forall x_i.\varphi$, for $i \in \{0,1\}$.

Clearly $\vdash_2$ is sound for first-order logic, but hopelessly incomplete. Trivial validities like $\forall x_0.(x_0 = x_0)$ and $\exists x_0 x_0.\varphi \leftrightarrow \exists x_0.\varphi$ are not theorems of $\vdash_2$.

THEOREM 1.13. [Areces and Marx, 1998] *For every k, there exist FO^2 formulas φ and ψ such that*

i. $\varphi \vdash_2 \psi$, *and*

ii. for every FO^k formula θ in the common language of φ and ψ, either $\varphi \not\models \theta$ or $\theta \not\models \psi$.

These formulas can be algorithmically obtained and have size polynomial in k. Either φ and ψ are in disjoint languages but both contain the equality symbol, or they are equality-free but the common language contains one binary predicate.

Even though it is difficult to put a finger on it, and in spite of the previous negative results, finite variable fragments seem better suited to our purposes than prenex normal form fragments. At least they seem to have better meta-logical properties than the prefix fragments. For instance, we can use the notion of k-back-and-forth-systems to provide a model-theoretical characterization of FO^k. Call a formula *invariant for k-back-and-forth systems* if and only if whenever there is a k-back-and-forth system linking two models $\mathcal{M}$ and $\mathcal{N}$ then $\mathcal{M} \models \varphi \Leftrightarrow \mathcal{N} \models \varphi$.

THEOREM 1.14. [van Benthem, 1995] *A first-order formula φ is equivalent to a formula in FO^k iff it is invariant for k-back-and-forth systems.*

And we can use a version of Ehrenfeucht games, called pebble-games (see [Immerman and Kozen, 1989; Ebbinghaus and Flum, 1999]), to obtain game-theoretical characterizations. Furthermore, modified versions of Łos-Tarski and Craig Theorems do go through (see [Andréka *et al.*, 1998]).

The main disadvantage of finite variable fragments, though, seems to be their coarseness: FO^1 is barely interesting, FO^2 is just fine, but most of its extensions aren't. With respect to decidability, FO^2 is not what Vardi [1997] has called *robustly decidable*, i.e., it does not remain decidable when suitable extensions of the language are considered. Also, k-variable fragments have a poor proof theory. No finitely axiomatized Hilbert style system exists [Monk, 1969], and the complexity of the necessary axiom schemes is inevitably high [Andréka, 1991].

1.2.3 Guarded Fragments

Clearly, the expressive power of FO comes from its quantifiers. In recent years, yet another way of defining fragments of first-order logic and its extensions was introduced by Andréka, van Benthem and Németi [1998], by imposing restrictions on the way a quantifier can be introduced in a formula.

Let GF, the *guarded fragment* of first-order logic, be defined as

- Every relational atomic formula $R(\bar{v})$ or $x_i = x_j$ belongs to GF.
- GF is closed under Boolean operations.
- If $\bar{v}$, $\bar{w}$ are tuples of variables, $\alpha(\bar{v}, \bar{w})$ is an atomic formula and φ is a formula in GF such that all free variables of φ occur in α, then also the formulas $\exists \bar{w}.(\alpha(\bar{v}, \bar{w}) \wedge \varphi(\bar{v}, \bar{w}))$ and $\forall \bar{w}.(\alpha(\bar{v}, \bar{w}) \rightarrow \varphi(\bar{v}, \bar{w}))$ belong to GF.

The intuitions behind GF are as follows: the new variables $\bar{w}$, introduced in $\varphi(\bar{v}, \bar{w})$ by the quantifiers are "bound" to a very simple syntactic condition (the atom α). The *guard* α constrains the part of the model that can be reached by the quantifier, thus making decidable the satisfiability problem of the fragment. These intuitions are very closely related to the techniques of algebraic relativization (see [Monk, 1993]). Notice an important property of the guarded fragment: it restricts neither the pattern of alternation of quantifiers as prenex normal form fragments do, nor the number of variables used as is done in FO^k.

In the same spirit, the *loosely guarded fragment* for first-order logic (LGF) is obtained by replacing the last clause in the definition of GF by a less strict one:

- If $\bar{v}$, $\bar{w}$ are tuples of variables, $\alpha(\bar{v}, \bar{w}) = \bigwedge \alpha_i$ is a conjunction of atoms and φ is a formula in LGF, then also the formulas $\exists \bar{w}.(\alpha(\bar{v}, \bar{w}) \wedge \varphi(\bar{v}, \bar{w}))$ and $\forall \bar{w}.(\alpha(\bar{v}, \bar{w}) \rightarrow$

$\varphi(\bar{v}, \bar{w})$) belong to LGF, provided that $\mathit{Free}(\varphi) \subseteq \mathit{Free}(\alpha)$, and for any two variables $z \in \bar{w}, z' \in \bar{v} \cup \bar{w}$ there is at least one atom α_i that contains both z and z'.

Grädel [2000], proved that the satisfiability problems for GF and LGF is complete for 2EXPTIME. Even further, the satisfiability problem for *guarded fixed point logic* ($\mu\mathsf{GF}$) is also complete for 2EXPTIME [Grädel and Walukiewicz, 1999], where $\mu\mathsf{GF}$ is obtained by adding to the definition of GF the clause

– Let R be a k-ary relation variable and let $\bar{v} = v_1, \ldots, v_k$ be a k-tuple of distinct variables. Let $\varphi(R, \bar{v})$ be a guarded formula where R appears only positively and not in guards and that contains no free variables other than $\bar{v}$. Then the formulas $[\mathsf{LFP}R\bar{v}.\varphi(\bar{v})]$ and $[\mathsf{GFP}R\bar{v}.\varphi(\bar{v})]$ are in $\mu\mathsf{GF}$.

$\mu\mathsf{GF}$ is actually a very powerful language, with expressivity beyond FO.

EXAMPLE 1.15. [Grädel, 1999] Let φ be the conjunction of the formulas

$$\exists x_0 x_1.R(x_0, x_1),$$
$$\forall x_0 x_1.(R(x_0, x_1) \to \exists x_0.R(x_1 x_0)),$$
$$\forall x_0 x_1.(R(x_0, x_1) \to [\mathsf{LFP}Sx_0.\forall x_1.(R(x_1, x_0) \to Sx_1)](x_0)).$$

The first two conjuncts force the model to have an infinite "forward" path through the R relation, while the third says that each point in the path is in the least fixed point of the operator $S \mapsto \{e \mid \text{all } R\text{-predecessors of } e \text{ are in } S\}$, which is the set of points that have only finitely many R-predecessors. In particular, this forces the path to be acyclic and hence the formula has only infinite models.

Many further results show that guarded fragments have nice behavior. Andréka *et al.* [1998] prove a Łos-Tarski theorem for preservation of guarded formulas under submodels, and also a model-theoretical characterization by means of a suitable definition of back-and-forth systems. Hoogland *et al.* [1999] provide the correct notions of interpolation and Beth definability for guarded fragments. Grädel and Walukiewicz' complexity results show that these fragments are, to a large extent, robustly decidable.

It is interesting to investigate how the intuitions built into the guarded fragments came to be. They are deeply rooted in modal languages.

1.3 Why Modal Logic?

Modal Logic [Blackburn *et al.*, 2000], originally conceived as the logic of necessity and possibility, has developed into a powerful mathematical discipline that deals with (restricted) description languages for talking about various kinds of relational structures. For many years, modal logic was viewed as an extension of propositional logic by the addition of the modal operators $\Diamond$ and $\Box$, but nowadays the picture has changed in many directions.

First, $\Diamond$ and $\Box$ have lost their privileged position, as a wide variety of new modalities have been introduced in recent years, witness for instance the work on Since-Until logics [Gabbay *et al.*, 1994], the universal modality [Goranko and Passy, 1992], the difference operator [de Rijke, 1992], propositional dynamic logic [Harel, 1984], counting modalities [de Rijke and van der Hoek, 1995], etc. Moreover, modal logic is no longer seen as just an extension of propositional logic, but also as a restriction of FO.

Again, it is better to do first things first and to start by introducing the basic modal language and its formal semantics.

DEFINITION 1.16. [Basic modal logic] Let $\mathsf{PROP} = \{p_1, p_2, \ldots\}$ be a countable set of *propositional variables.* The *well-formed formulas* of the basic modal language over PROP are

$$\mathsf{FORMS} := \top \mid p_i \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \Diamond\varphi,$$

where $p_i \in \mathsf{PROP}$ and $\varphi, \varphi_1, \varphi_2 \in \mathsf{FORMS}$. We take $\Box\varphi$ as a shorthand for $\neg\Diamond\neg\varphi$.

A *modal model* $\mathcal{M}$ for $\mathcal{L}$ is a triple $\mathcal{M} = \langle M, R, V\rangle$ such that M is a non-empty set, R a binary relation on M, and $V : \mathsf{PROP} \to Pow(M)$. Let $\mathcal{M} = \langle M, R, V\rangle$ be a model and $m \in M$, then the *satisfiability relation* $\Vdash$ is defined as

$$\begin{array}{rll}
\mathcal{M}, m \Vdash \top & & \text{always} \\
\mathcal{M}, m \Vdash p_i & \text{iff} & m \in V(p_i),\ p_i \in \mathsf{PROP} \\
\mathcal{M}, m \Vdash \neg\varphi & \text{iff} & \mathcal{M}, m \nVdash \varphi \\
\mathcal{M}, m \Vdash \varphi_1 \wedge \varphi_2 & \text{iff} & \mathcal{M}, m \Vdash \varphi_1 \text{ and } \mathcal{M}, m \Vdash \varphi_2 \\
\mathcal{M}, m \Vdash \Diamond\varphi & \text{iff} & \text{for some } m' \in M \text{ such that } R(m, m') \text{ holds, } \mathcal{M}, m' \Vdash \varphi.
\end{array}$$

From the definition it is clear that the language can be seen as an extension of propositional logic. But we can now make precise our remark that it is also a fragment of first-order logic. Notice that a modal model $\langle M, R, V\rangle$ can be seen as a first-order model by considering V as the part of the interpretation function defining the meaning of unary predicate symbols. Also, the conditions defining $\Vdash$ are purely first-order. Putting these two intuitions together we obtain the following embedding.

DEFINITION 1.17. [Standard translation] Consider the signature $\mathcal{S} = \langle\{R\} \cup \{P_i \mid p_i \in \mathsf{PROP}\}, \{\}, \{\}\rangle$. And define the translation ST from modal formulas to first-order formulas over $\mathcal{S}$ as follows:

$$\begin{array}{rcl}
ST(p_j) & = & P_j(x_0),\ p_j \in \mathsf{PROP} \\
ST(\neg\varphi) & = & \neg ST(\varphi) \\
ST(\varphi \wedge \psi) & = & ST(\varphi) \wedge ST(\psi) \\
ST(\Diamond\varphi) & = & \exists x_j.(R(x_0, x_j) \wedge ST(\varphi)[x_0/x_j]),
\end{array}$$

where j is the smallest index such that x_j does not appear in $ST(\varphi)$.

THEOREM 1.18. [Satisfiability preservation] *Let φ be a formula in the modal language, then for any model $\mathcal{M}$ in the appropriate signature, $\mathcal{M}, m \Vdash \varphi$ iff $\mathcal{M} \models ST(\varphi)[m]$.*

Performing the translation with slightly more care and reusing variables, we can actually embed a basic modal formula φ into FO^2.

DEFINITION 1.19. [Standard translation into FO^2] In [Vardi, 1997] the following translation is introduced

$$
\begin{array}{rcl}
ST(p_j) & = & P_j(x_0),\ p_j \in \mathsf{PROP} \\
ST(\neg\varphi) & = & \neg ST(\varphi) \\
ST(\varphi \wedge \psi) & = & ST(\varphi) \wedge ST(\psi) \\
ST(\Diamond\varphi) & = & \exists x_1.(R(x_0, x_1) \wedge \forall x_0.(x_0 = x_1 \rightarrow ST(\varphi))),
\end{array}
$$

but this uses a more complicated pattern of quantification and also identity. We can do even better by using a pair of mutually recursive functions.

$$
\begin{array}{r|r}
ST_{x_0}(p_j) = P_j(x_0),\ p_j \in \mathsf{PROP} & ST_{x_1}(p_j) = P_j(x_1),\ p_j \in \mathsf{PROP} \\
ST_{x_0}(\neg\varphi) = \neg ST_{x_0}(\varphi) & ST_{x_1}(\neg\varphi) = \neg ST_{x_1}(\varphi) \\
ST_{x_0}(\varphi \wedge \psi) = ST_{x_0}(\varphi) \wedge ST_{x_0}(\psi) & ST_{x_1}(\varphi \wedge \psi) = ST_{x_1}(\varphi) \wedge ST_{x_1}(\psi) \\
ST_{x_0}(\Diamond\varphi) = \exists x_1.(R(x_0, x_1) \wedge ST_{x_1}(\varphi)) & ST_{x_1}(\Diamond\varphi) = \exists x_0.(R(x_1, x_0) \wedge ST_{x_0}(\varphi)).
\end{array}
$$

Clearly, Theorem 1.18 can also be proved for these new translations. And notice that we have immediately obtained two different proofs of the decidability of the satisfiability problem for the basic modal logic, as the range of ST is both inside FO^2 and GF. Actually, robust decidability obtains, as even the μ-calculus [Kozen, 1983] (the extension of the basic modal language with fixed points) has an exponential-time decision procedure for satisfiability (a double exponential upper bound can easily be obtained via translation into $\mu\mathsf{GF}$).

The embedding of the modal language into the guarded fragments comes as no surprise, as these fragments were actually first thought of as generalizations of the basic modal language. But "generalizations" are at odds with our aim of obtaining specialized, hand-tailored languages. Modal languages instead, can be used *themselves* as ways to identify interesting fragments of first-order logic, and even of extensions as is shown by examples such as the μ-calculus mentioned above, Propositional Dynamic Logic [Harel, 1984], or the Computational Tree Logic of Clarke and Emerson [1982].

Some attractive properties are often shared by modal languages: a propositional flavor, a simple and uniform semantics, elegant completeness results, strong decidability, interpolation and Beth definability, effective reasoning systems, model-theoretical characterizations, adequate tools to compare models. The list is just too long to be given in full. In other words, using modal languages as a means to define well behaved fragments of first-order logic and extensions seems a promising project.

1.4 Some Concrete Examples

Example 1.20. Suppose we want to capture the information conveyed in the following paragraph:

> *'And there is one among them that might have been foaled in the morning of the world. The horses of the Nine cannot vie with him; tireless, swift as the flowing wind. Shadowfax they called him. By day his coat glistens like silver; and by night it is like a shade, and he passes unseen.'*
>
> from "The Lord of the Rings," J. R. R. Tolkien

We can think of the domain of models as "the set of all things" in Tolkien's world and define certain relations among them, like for example the unary relation "tireless-things" which consist of the set of things that never tire, or the binary relation "has-coat-like" which relates a thing a with a thing b if a happens to have a coat that looks like b. "Shadowfax" is, of course, one of the elements in the domain, and we can define a special unary relation symbol `shadowfax`, whose denotation is the singleton subset of the domain containing it. Consider the following formulas,

$$\texttt{shadowfax} \to \left(\begin{array}{c} \langle \textit{foaled} \rangle \text{ things-existing-in-the-morning-of-the-world} \wedge \\ \text{tireless-things} \ \wedge \ \text{swift-as-the-wind-things} \wedge \\ [\textit{has-coat-like}] \text{ things-glistening-like-silver-by-day} \wedge \\ [\textit{has-coat-like}] \text{ things-like-shades-by-night} \ \wedge [\textit{pass-like}] \text{ unseen-things} \end{array} \right)$$

$$\text{horses-of-the-Nine} \to \neg \langle \textit{vie} \rangle \ \texttt{shadowfax}.$$

They seem to capture a good deal of the information contained in the text. And we can use some of the extensions to the basic modal language we already mentioned, to define very expressive concepts. For example, if the "purest-breed" are those horses sired by Shadowfax or his descendants, using fixed points we define

$$\text{purest-breed} \leftrightarrow \mathsf{LFP}X.(\texttt{shadowfax} \vee [\textit{sired-by}]X).$$

EXAMPLE 1.21. From a different angle, suppose we want to describe a set of structures, like for example, full binary trees. We can define in the basic modal language, for any $n \in \mathbb{N}$ a formula φ_n such that

- the size of φ_n is quadratic in n,
- φ_n is satisfiable,
- if $\mathcal{M}, w \Vdash \varphi_n$, then $\mathcal{M}$ contains as a substructure an isomorphic copy of the binary tree of depth n with root w.

We fix a set $\{p_0, \ldots, p_{n-1}\}$ of propositional symbols, and start by defining the shorthands $\textit{branch}(p_i) := \Diamond p_i \wedge \Diamond \neg p_i$ and $\textit{store}(p_i) := (p_i \to \Box p_i) \wedge (\neg p_i \to \Box \neg p_i)$. Then

$$\varphi_n := \textit{branch}(p_0) \wedge \bigwedge_{1 \le i < n} \Box^i (\textit{branch}(p_i) \wedge \bigwedge_{0 \le j < i} \textit{store}(p_j)),$$

in which $\Box^i$ abbreviates an i-long sequence of boxes. The formula works as follows. Suppose $\mathcal{M}, m \Vdash \varphi_n$. Then, because $\Box^i \textit{branch}(p_i)$ is satisfied, every node m' reachable in i R-steps from m has two different successors, one forcing p_i and one forcing $\neg p_i$. But in each R-step we used the *store* formula to ensure that the value of p_j for $j < i$ is "carried over" into the next state. It is easy to check that the interplay of *branch* and *store* forces a binary tree of depth n. As an aside, notice that this implies that the size of the smallest model satisfying φ_n is exponential in $|\varphi_n|$. That the basic modal language is expressive enough to force big models is a well known fact, see [Marx and Venema, 2000] for example, for an up-to-date survey on the topic.

Modal languages are very well suited to describe graphs as we did above and this, together with their effective decision methods, produces a powerful combination.

In [Areces *et al.*, 1999b] for example, we use modal languages to encode the transition diagrams of simple telephony protocols, together with specific *telephony features* — special telephony services like Call Waiting or Call Forwarding. We then formally verify that the behavior of a feature does not interfere with the behavior of others in an unexpected way, by means of automatic tools.

Example 1.22. Believe it or not, modal logics can also help you to properly set a table for a formal dinner. Aiello and van Benthem [1999] discuss modal languages to describe topological structures. In this work, topological spaces are used instead of relational models and the basic modalities $\Diamond$ and $\Box$ are interpreted as the interior and the closure of a region, respectively. When further expressive power is added, the language can express *betweenness* and hence distinguish between the well set table on the left and the messy one on the right in Figure 1.1.

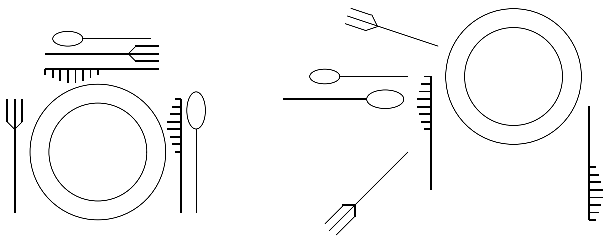

Figure 1.1: Properly setting a table

In [Aiello *et al.*, 1999] we use similar ideas to propose an image description language and an image retrieval engine. Topological operators are used to describe the relations between regions in an image. In addition, further information can be naturally added, similarly as how we did in Example 1.20. A query to the image database can then be recast as a formula, and the retrieval process transformed into inference: given a query φ retrieve all pictures P such that their description δ_P implies the query, $\models \delta_P \to \varphi$.

1.5 This Thesis

The topics discussed in this chapter should give the reader a taste of what is to come. As we mentioned before, we will investigate modal systems (broadly conceived) as a way to define well behaved fragments. In particular, we will investigate the family of systems known as Description Logics which are probably one of the best examples of "application-driven" formal languages.

Description logics form a collection of languages that come equipped with effective inference methods. These languages were born in the Knowledge Representation community, and this shows in the mixture of theoretical and practical results which is almost a hallmark of the literature in the field. These languages are closely related to the basic modal language, but they usually add further expressivity depending on the specific problem at hand. Systems containing operators like counting, transitive closure, fixed points or nominals, to mention some, have been proposed and widely used. In each case, effective decision methods have been devised and their complexity carefully analyzed. On the other hand, and probably as a result of the fact that many of these languages were introduced to tackle a particular problem, an *unifying* logical background theory to permit for example, the comparison of their respective expressive power, is largely missing. In this thesis, we will put forward the claim that *hybrid logics*, extensions of the basic modal language with the ability to explicitly refer to elements in the domain, provide such a unifying framework.

The themes we discussed in this introduction are at the root of the work we will do throughout the thesis. If you survived up to here, you are ready to go on. Theorem 1.1 focused on undecidability and, more generally on complexity of reasoning problems. For first-order logic we have only discussed satisfiability and validity, but when moving to modal languages we will see different examples of reasoning tasks, in addition to these two. We will also touch on different reasoning methods like tableau and resolution calculi, and on the advantages and disadvantages of each of them, both from a theoretical point of view and in implementations. In Theorem 1.2 we discussed matters related to expressive power. This is a core issue, as little is known about ways of comparing the expressive power of description languages. The tools here are model comparison techniques such as the Ehrenfeucht games, and the central notion of bisimulation (the modal counterpart of partial isomorphisms). Finally, with Theorem 1.13 we touched on particular meta-logical properties like interpolation and Beth definability, which are usually a sign of a "healthy" reasoning system.

Part II of the thesis, starts by introducing the two different families of languages we will study, and concludes by building a bridge between them. Chapter 2 will cover Description Logics, while Chapter 3 will introduce Hybrid Logics. Chapter 4, which is based on early results in [Areces and de Rijke, 1998], and on the more up to date [Areces and de Rijke, 2000], establishes the logical correlation between the two different frameworks. In this chapter we show the strength of the connection between description and hybrid languages, and exemplify in detail how results in one of the fields yield interesting facts in the other.

The three chapters in Part III form the core of the thesis, each of them channeling results into the logical connections we discuss in Chapter 4. In [Areces *et al.*, 1999c], we present a resolution system for modal and description languages. These results are discussed in Chapter 5, and the resolution calculus is extended to deal with hybrid binders. In Chapter 6 we present the characterization results obtained in [Areces *et al.*, 2000b], together with interpolation and definability results for hybrid systems. Chapter 7 is devoted to (un)decidability and complexity issues. The articles [Areces *et al.*, 2000a] and [Areces *et al.*, 1999a] are concerned with sharp undecidability results (what is the borderline between decidability and undecidability for very expressive hybrid systems?), and with the effect that "hybridization" (the addition of the particular ability to refer to states) produces in modal languages from the complexity point of view. These results are related to the already known results for certain description languages. The key results discussed in Part III of the thesis were obtained in collaboration with Patrick Blackburn, Maarten Marx, Hans de Nivelle and Maarten de Rijke. Not only did I learn a great deal from them, but also I enjoyed myself immensely during this joint work, and I can hardly thank them enough for all the help I've received from them.

Finally, Part IV concludes with Chapter 8 where we draw our conclusions and point to directions of further research.

Chapter 2

Introducing Description Logics

Esas ambigüedades, redundancias y deficiencias recuerdan las que el doctor Franz Kuhn atribuye a cierta enciclopedia china que se titula Emporio celestial de conocimientos benévolos. *En sus remotas páginas está escrito que los animales se dividen en (a) pertenecientes al Emperador, (b) embalsamados, (c) amaestrados, (d) lechones, (e) sirenas, (f) fabulosos, (g) perros sueltos, (h) incluidos en esta clasificación, (i) que se agitan como locos, (j) innumerables, (k) dibujados con un pincel finísimo de pelo de camello, (l) etcétera, (m) que acaban de romper el jarrón, (n) que de lejos parecen moscas.*

from "El Idioma Analítico de John Wilkins," Jorge Luis Borges

2.1 Structure: the Key to Knowledge

Knowledge about anything abounds. For a test, think of any single word and run a query on a search engine on the Internet. It doesn't matter which word, you will probably get (at least) hundreds of hits. The problem of course, is that we are usually not interested in *any kind* of knowledge, and probably only few of the hundreds of hits are actually relevant to us. In other words, information should be structured to be useful, so that we can decide which part of it is important to our problem. But classifying information is a difficult and expensive task.

Structured representation of knowledge aims to address both conceptual and computational complexity. Conceptual economy amounts to building hierarchical structures, where inheritance of attributes through the hierarchy is used to avoid redundancies in the representation. Computational economy refers to the efficiency of reasoning upon such structures. The idea of developing systems based on a structured representation of knowledge has been pursued for a long time in Artificial Intelligence. One of the earliest knowledge representation tools has been the *Semantic Networks*, and the work of Quillian [1967] on the *Semantic Memory Model.* Semantic networks represent knowledge in the form of a labeled directed graph. Specifically, each node is associated with a concept, and the arcs represent the various relations between concepts.

In a similar line, the work of Minsky [1974] on the *Frame Paradigm*, aims to achieve structured knowledge. A frame represents a certain concept (usually a class of individuals), and is characterized by a number of attributes (called *slots*) that members of its class can have. Each slot contains information about the corresponding attribute, such as default values, restrictions on the elements that can fill the attribute (*slot fillers*), attached procedures or methods for computing values when needed, and procedures for propagating side effects when the slot is filled. The values of the attributes are either elements of a concrete domain (e.g., integers, strings) or identifiers of other frames. A frame can also represent a single individual, in this case it is related with the attribute *instance-of* to the frame representing the class of which the individual is an instance.

The main drawback of both proposals was their unclear semantics. As pointed out in [Woods, 1975] with respect to semantic networks, the arcs between the nodes and the nodes themselves can represent different kinds of information, and making this difference explicit can be crucial. Consider the following example.

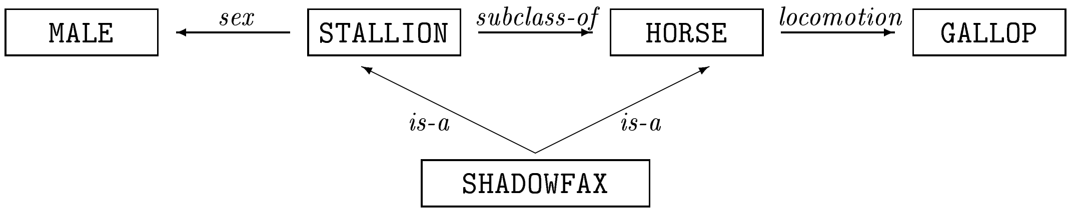

Figure 2.1: Shadowfax the stallion

The network is meant to encode the following information: Shadowfax is a stallion and also a horse, stallions are male, horses gallop, and stallions are horses.

Woods defines two main types of arcs, that he identifies as encoding either *intensional* or *extensional* information. An arc that contributes to the definition of a concept carries intensional knowledge. The arc labeled *locomotion* between `HORSE` and `GALLOP` is of this kind. On the other hand, the arc from `SHADOWFAX` to `STALLION`, labeled *is-a*, asserts the fact that Shadowfax is a stallion, which Woods classifies as extensional knowledge. But things are more subtle, because also the nodes carry different information: `STALLION` is a class, while `SHADOWFAX` is a distinguished individual. And furthermore, the "intensional" relation between `STALLION` and `HORSE` is of a different kind than the relation represented by *locomotion*. The distinction between extensional and intentional links, neither exhausts nor characterizes all the possibilities.

The need for a formal semantics was clear, and [Brachman, 1977, 1979] are some of the early references which aimed to address this problem. The work of Brachman led to the development of KL-ONE [Brachman and Schmolze, 1985], one of the first knowledge maintainance systems for which some kind of formal semantics was specified. The KL-ONE system fathered a number of successors such as KRYPTON [Brachman *et al.*, 1985], LOOM [MacGregor and Bates, 1987], CLASSIC [Borgida *et al.*, 1989], BACK [Quantz and Kindermann, 1990] and KRISS [Baader and Hollunder, 1991]. But, perhaps more importantly, this work also gave rise to theoretical research on what were first named *concept* or *terminological languages*, and which are today called *description logics*.

Description logics (DLs) are a family of formal languages with a clearly specified semantics, usually in terms of first-order models, together with specialized inference mechanisms to account for knowledge classification. It was clearly one of the original and main aims of the research in this field to identify the exact fragments of FO and extensions able to capture the features needed for representing a particular problem, and which can still allow for the design of efficient reasoning algorithms.

DLs have found a variety of applications in diverse areas. To mention some, [Borgida, 1995] is a survey of the application of DLs to the problems of information management, proposing ways to achieve enhanced access to data and knowledge by using descriptions in languages for schema design and integration, queries, updates, rules, and constraints.

Another example is [Devanbu and Jones, 1997]. The increasing size and complexity of software systems demand every day a greater emphasis on capturing and maintaining knowledge at different levels within the software development process. The knowledge-based software engineering (KBSE) paradigm is concerned with systems that use formally represented knowledge, with associated inference procedures, to support the various sub-activities of software development. Devanbu and Jones investigate the application of DLs to KBSE, describing their use in three well-developed KBSE systems: LaSSIE and KITSS in the telephony domain and COMET in the radar tracking domain.

PROSE (PRoduct OfferingS Expertise) is a knowledge based engineering and ordering platform that supports sales and order processing at AT&T Network Systems. The cornerstone of the PROSE architecture is a product knowledge base written in C-CLASSIC. PROSE is used to provide configurations for sales proposals and to generate factory orders for manufacturing. A fairly detailed description of the system can be found in [Wright *et al.*, 1993].

DLs have also been used extensively for general information retrieval as is described in [Meghini *et al.*, 1993] where the system MIRTL (Multimedia Information Retrieval Terminological Logic) is introduced. In [Aiello *et al.*, 1999] we investigated special mixed description languages which coordinate satisfiability and model checking to provide expressive image retrieval which can account for topological relations between objects in pictures. As a final example, in [Areces *et al.*, 1999b] we use description languages to model BCS, the Basic Call System used in telephony domains and formally investigate the issue of feature interaction, analyzing the problems arising when merging new services like Call Waiting, Call Forwarding, etc. DLs have even found their way into general, school level Artificial Intelligence, and they are already discussed in text books like [Russell and Norvig, 1995].

2.2 Basic Issues in Description Logic

Probably rooted in the original distinction of Woods, most description languages split the available knowledge about a given situation into

- *terminological information*: definitions of the basic and derived notions and of the ways they are inter-related. This information is "generic" or "global," been true in every model of the situation and of every individual in the situation. And
- *assertional information*: which records "specific" or "local" information, being true of certain particular individuals in the situation.

All known information is then modeled as a pair $\langle T, A\rangle$, where T is a set of formulas concerning terminological information (the T-Box) and A is a set of formulas concerning assertional information (the A-Box).

Another way to look at this separation of information is from a database point of view: the T-Box is a general schema concerning the classes of individuals to be represented, their general properties and mutual relationships, while the A-Box is a partial instantiation of this schema, containing assertions relating either individuals to classes, or individuals to each other.

Constructor	Syntax	Semantics
concept name	C	$C^{\mathcal{I}}$
top	$\top$	$\Delta^{\mathcal{I}}$
negation ($\mathcal{C}$)	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
disjunction ($\mathcal{U}$)	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
universal quant.	$\forall R.C$	$\{d_1 \mid \forall d_2 \in \Delta^{\mathcal{I}}.(R^{\mathcal{I}}(d_1, d_2) \rightarrow d_2 \in C^{\mathcal{I}})\}$
existential quant. ($\mathcal{E}$)	$\exists R.C$	$\{d_1 \mid \exists d_2 \in \Delta^{\mathcal{I}}.(R^{\mathcal{I}}(d_1, d_2) \wedge d_2 \in C^{\mathcal{I}})\}$
number restr. ($\mathcal{N}$)	$(\geq n\ R)$	$\{d_1 \mid \lvert\{d_2 \mid R^{\mathcal{I}}(d_1, d_2)\}\rvert \geq n\}$
	$(\leq n\ R)$	$\{d_1 \mid \lvert\{d_2 \mid R^{\mathcal{I}}(d_1, d_2)\}\rvert \leq n\}$
one-of ($\mathcal{O}$)	$\{a_1, \ldots, a_n\}$	$\{d \mid d = a_i^{\mathcal{I}}$ for some $a_i\}$
role filler ($\mathcal{B}$)	$\exists R.\{a\}$	$\{d \mid R^{\mathcal{I}}(d, a^{\mathcal{I}})\}$
role name	R	$R^{\mathcal{I}}$
role conjunction ($\mathcal{R}$)	$R_1 \sqcap R_2$	$R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}}$
inverse roles ($\mathcal{I}$)	R^{-1}	$\{(d_1, d_2) \mid R^{\mathcal{I}}(d_2, d_1)\}$

Table 2.1: Common operators of description logics

2.2.1 Concepts and Roles

Let us make things more precise now.

Definition 2.1. [Description logic semantics] Let $\mathsf{CON} = \{C_1, C_2, \ldots\}$ be a countable set of *atomic concepts*, $\mathsf{ROL} = \{R_1, R_2, \ldots\}$ be a countable set of *atomic roles* and $\mathsf{IND} = \{a_1, a_2, \ldots\}$ be a countable set of *individuals*. For $\mathsf{CON}, \mathsf{ROL}, \mathsf{IND}$ pairwise disjoint, $\mathcal{S} = \langle \mathsf{CON}, \mathsf{ROL}, \mathsf{IND} \rangle$ is a *signature*. Once a signature $\mathcal{S}$ is fixed, an *interpretation* $\mathcal{I}$ for $\mathcal{S}$ is a tuple $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where

- $\Delta^{\mathcal{I}}$ is a non empty set.
- $\cdot^{\mathcal{I}}$ is a function assigning an element $a_i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ to each constant a_i; a subset $C_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to each atomic concept C_i; and a relation $R_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to each atomic role R_i.

In other words, a description logic interpretation is no more than a model for a particular kind of first-order signature (see Definition 1.4), where only unary and binary predicate symbols are allowed and the set of function symbols is empty.

The atomic symbols in a description logic signature can be combined by means of *concept and roles constructors*, to form complex concept and role expressions. Each description logic is characterized by the set of concept and roles constructors they allow. Table 2.1 defines the roles and concepts constructors for the description logics we will discuss, together with their semantics.

Historically, a number of description logics have received a special name. The language $\mathcal{FL}^-$ [Brachman and Levesque, 1984] is defined as the description logic allowing universal quantification, conjunction and unqualified existential quantifications of the form $\exists R.\top$. $\mathcal{FL}^-$ was proposed as a formalization of the core notions of Minsky's frames. Concept conjunction is implicit in the structure of a frame, which requires a *set* of conditions to be satisfied. Role quantifications allow one to characterize slots: the unqualified existentials state the existence of a value for a slot, while the universal quantifier requires that the values of a slot satisfies a certain condition.

The logic $\mathcal{AL}$ [Schmidt-Schauss and Smolka, 1991] extends $\mathcal{FL}^-$ with negation of atomic concepts. It is customary to define systems by postfixing the names of these original systems with the names of the added operators from Table 2.1. For example, the logic $\mathcal{ALC}$ is $\mathcal{AL}$ extended with full negation.

We will not discuss in detail all possible languages which can be obtained by combining constructors from Table 2.1. In particular, we will be interested in languages having full Boolean expressivity and usually consider $\mathcal{ALC}$ and its extensions. For a given language $\mathcal{L}$, let $\mathsf{CON}(\mathcal{L})$ be the set of complex concept expressions and $\mathsf{ROL}(\mathcal{L})$ be the set of complex role expressions which can be formed by using the constructors of $\mathcal{L}$. It is interesting to notice that the constructs in Table 2.1 are not necessarily independent of each other. Given a language $\mathcal{L}$ and a constructor $\star$, we say that $\mathcal{L}$ *simulates* $\star$ if for every complex concept of $\mathcal{L}\star$ there exists an equivalent concept in $\mathcal{L}$. Formally, for any $C_1 \in \mathsf{CON}(\mathcal{L}\star)$, there exists $C_2 \in \mathsf{CON}(\mathcal{L})$ such that for all interpretations $\mathcal{I}$, $C_1^{\mathcal{I}} = C_2^{\mathcal{I}}$. Given this definition, proving for example that $\mathcal{ALEU}$ simulates $\mathcal{C}$ or that $\mathcal{ALEO}$ simulates $\mathcal{B}$ is straightforward, by means of simple reductions. We will usually assume that all constructors which can be simulated in $\mathcal{L}$ are already present in the language, e.g., we will say that $\mathcal{U}$ is one of the constructors of $\mathcal{ALC}$.

2.2.2 Knowledge Bases and Inference

In description logics we want to perform inferences given certain background knowledge.

DEFINITION 2.2. [Knowledge bases] Fix a description language $\mathcal{L}$, a *knowledge base* Σ in $\mathcal{L}$ is a pair $\Sigma = \langle T, A \rangle$ such that

- T is the T(erminological)-Box, a finite, possibly empty, set of expressions of the form $C_1 \sqsubseteq C_2$ where C_1, C_2 are in $\mathsf{CON}(\mathcal{L})$. $C_1 \doteq C_2$ is notation for $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$. Formulas in T are called *terminological axioms*.
- A is the A(ssertional)-Box, a finite, possibly empty, set of expressions of the forms $a\!:\!C$ or $(a, b)\!:\!R$ where C is in $\mathsf{CON}(\mathcal{L})$, R is in $\mathsf{ROL}(\mathcal{L})$ and a, b are individuals. Formulas in A are called *assertions*.

The definitions of terminological axioms and assertions above are among the most general in the description logic literature (and we will generalize them even further in Chapter 4). Terminological axioms were originally thought of as *definitions*, and a number of more restrictive conditions were imposed. The two most important restrictions were the following.

i. Simple terminological axioms: in any terminological axiom $C_1 \sqsubseteq C_2$, C_1 is an atomic concept in CON. And any atomic concept in CON appears at most once in the left hand side of a terminological axiom in the T-Box.

ii. Acyclic definitions: the graph obtained by assigning a node n_A to each atomic concept A in the T-Box T and drawing an arrow between two nodes n_A and n_B if there is a terminological axiom $C_1 \sqsubseteq C_2$ in T such that A appears in C_1 and B appears in C_2, does not contain cycles.

These restrictions were rooted in the idea of considering terminological axioms as (partial) definitions of concepts. An axiom of the form $A \sqsubseteq C$ was meant to represent the fact that C encoded necessary conditions for A to be the case, $A \doteq C$ indicated that C encoded both necessary and sufficient conditions. From this point of view, conditions *i*) and *ii*) above are very natural. In particular, restriction *ii*) was aimed at avoiding circular definitions, where a concept is defined in terms of itself. It was argued that this kind of definitions called for some sort of special semantics, like for example fixed points or non-well-founded sets [Nebel, 1990a; Baader, 1990; Dionne *et al.*, 1992]. But general terminological axioms as introduced in Definition 2.2 act naturally as constraints on models of the knowledge base, without any need for a special semantics; this way of interpreting terminologies is usually called "descriptive semantics" in the DL literature.

More importantly from an application point of view, restrictions *i*) and *ii*) have a strong impact on the reducibility of certain reasoning tasks into others as we will discuss in Section 2.3. Clearly, this kind of syntactic restrictions undercut the expressive power of the language, but they simplify the definition of decision algorithms. As we discuss in Chapter 4, the complexity results we prove in Chapter 7 show that in many cases we can actually introduce a more general notion of knowledge base than the one given in Definition 2.2 without modifying the worst case complexity of the reasoning tasks for the language.

It is time to define the appropriate notion of inference for description logics.

Definition 2.3. Let $\mathcal{I}$ be an interpretation and φ a terminological axiom or an assertion. Then $\mathcal{I}$ *models* φ (notation, $\mathcal{I} \models \varphi$) if

- $\varphi = C_1 \sqsubseteq C_2$ and $C_1^{\mathcal{I}} \subseteq {C_2}^{\mathcal{I}}$, or
- $\varphi = a{:}C$ and $a^{\mathcal{I}} \in C^{\mathcal{I}}$, or
- $\varphi = (a, b){:}R$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

Let $\Sigma = \langle T, A\rangle$ be a knowledge base and $\mathcal{I}$ an interpretation, then $\mathcal{I}$ models Σ (notation, $\mathcal{I} \models \Sigma$) if for all $\varphi \in T \cup A, \mathcal{I} \models \varphi$. We say in this case that $\mathcal{I}$ *is a model of the knowledge base* Σ. Given a knowledge base Σ and a terminological axiom or assertion φ, $\Sigma \models \varphi$ if for all models $\mathcal{I}$ of Σ we have $\mathcal{I} \models \varphi$.

The notion of semantic consequence we have just defined, $\Sigma \models \varphi$, is central in modern logic. Notice that if a deduction theorem fails for the logic, it needs not be reducible to "tautology hood," i.e., to a given formula in the language being satisfied under any interpretation $\mathcal{I}$. We will return to this in Section 4.2.1.

2.3 Reasoning Tasks

In description logics the term *T-Box reasoning* is used to refer to the ability to perform inferences from a knowledge base $\Sigma = \langle T, A\rangle$ where T is non-empty, and similarly, *A-Box reasoning* is inference for A non-empty.

Example 2.4. Let Σ be a knowledge base $\langle T, A\rangle$ where

$$\begin{array}{rcl} T & = & \{\texttt{STALLION} \doteq \texttt{HORSE} \sqcap \forall \texttt{Sex}.\texttt{MALE}\} \\ A & = & \{\texttt{shadowfax}{:}\texttt{STALLION}\}. \end{array}$$

Informally, the formula in T says that male horses are called stallions, while the formula in A says that a particular horse, named Shadowfax, is a stallion. The formal semantics we gave in Definition 2.3 lets us verify that Σ has at least a model (i.e., it is consistent). And from Σ we can infer further information like, for example, that the concept HORSE is consistent with respect to Σ (there exists some interpretation satisfying Σ which assigns a non-empty extension to HORSE:

$$\Sigma \not\models \texttt{HORSE} \doteq \bot.$$

Notice though, that we cannot express the stronger (true) fact that in any model of $\langle T, A\rangle$, the extension of HORSE is non-empty.

$$\Sigma \models \neg(\texttt{HORSE} \doteq \bot),$$

because of syntactic limitations in the standard definition of assertions. In Chapter 4 we will further discuss this issue, and clarify why $\Sigma \models \neg(\texttt{HORSE} \doteq \bot)$ is not equivalent to $\Sigma \not\models \texttt{HORSE} \doteq \bot$.

We can actually map out a set of *reasoning tasks* or *reasoning services* which can be provided by a knowledge representation system. The following are some of the standard reasoning tasks usually considered in description logics.

DEFINITION 2.5. [Reasoning tasks] Let Σ be a knowledge base, $C_1, C_2 \in \mathsf{CON}(\mathcal{L})$, $R \in \mathsf{ROL}(\mathcal{L})$ and $a, b \in \mathsf{IND}$, we define the following *reasoning tasks*

- *Subsumption*, $\Sigma \models C_1 \sqsubseteq C_2$.
  Check whether for all interpretations $\mathcal{I}$ such that $\mathcal{I} \models \Sigma$ we have $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.
- *Instance Checking*, $\Sigma \models a\!:\!C$.
  Check whether for all interpretations $\mathcal{I}$ such that $\mathcal{I} \models \Sigma$ we have $a^{\mathcal{I}} \in C^{\mathcal{I}}$.
- *Relation Checking*, $\sigma \models (a, b)\!:\!R$.
  Check whether for all interpretations $\mathcal{I}$ such that $\mathcal{I} \models \Sigma$ we have $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.
- *Concept Consistency*, $\Sigma \not\models C \doteq \bot$.
  Check whether for some interpretation $\mathcal{I}$ such that $\mathcal{I} \models \Sigma$ we have $C^{\mathcal{I}} \neq \{\}$.
- *Knowledge Base Consistency*, $\Sigma \not\models \bot$.
  Check whether there exists $\mathcal{I}$ such that $\mathcal{I} \models \Sigma$.

Subsumption is one of the classical reasoning tasks performed in most knowledge representation systems. It is directly related to the quest for information classification: it is useful to organize concepts into "is-a" hierarchies, finding for each class the most specific other classes that subsume it. This classification algorithms relies directly on the subsumption check. The subsumption relation implicitly defines a taxonomy of concepts, which can actually be used to solve some of the other reasoning tasks; for example, a concept is unsatisfiable if it is subsumed by the empty concept $\bot$.

Instance checking is used to verify whether the knowledge base entails that an individual is an instance of a concept, while relation checking determines if two individuals in the knowledge base stand in a given relation. These two operations can be considered the central reasoning tasks for retrieving information about individuals from a knowledge base. Finally, consistency is used for verifying whether the information contained in a knowledge base is coherent.

As we already said, the basic reasoning tasks can be used to define more complex ones. In particular

- *Retrieval*: given a concept, find all the individuals mentioned in the knowledge base that are instances of the concept.
- *Realization*: given an individual mentioned in the knowledge base, find the most specific concepts, with respect to the subsumption relation, of which the individual is an instance.

Let's start by defining the following notation. For any set S of formulas let $\mathsf{IND}(S)$ and $\mathsf{CON}(S)$ be, respectively, the set of individuals and atomic concepts appearing in formulas in S. Then, the retrieval problem can be formulated as follows: given a knowledge base Σ and a concept C, find the set $\{a \in \mathsf{IND}(\Sigma) \mid \Sigma \models a:C\}$; and it can be performed simply by iterating instance checking for all the individuals in Σ. The realization problem can be solved by finding the set $\{C \in \mathsf{CON}(\Sigma) \mid \Sigma \models a:C \;\&\; \forall C' \in \mathsf{CON}(\Sigma).(\Sigma \models a:C' \Rightarrow \Sigma \models C \sqsubseteq C')\}$; this can be done in terms of instance checking and subsumption.

Research on description logics has focused mainly on understanding the relations between the reasoning tasks mentioned above, and on establishing their computational complexity. The study of the computational behavior of DLs has provided a good understanding of the properties of the language constructs and their interaction. This is not only valuable from a theoretical point of view, but also provides insight to the designer of deduction procedures, with indications of which language constructs are difficult to handle and general methods to cope with the computational problems.

Complexity analyses of subsumption originated with the seminal paper of Brachman and Levesque [1984], where they provide a polynomial algorithm to decide subsumption for $\mathcal{FL}^-$. In the early days of description logics, the emphasis was on mapping out the *tractable* reasoning tasks, with an upper bound of polynomial complexity. Clearly, only very weak (non-Boolean) languages were able to survive the test, as already the satisfiability problem of mere propositional logic is NP-complete. The $\mathcal{U}$ and $\mathcal{E}$ constructors were identified as the main culprits of "intractability," the first one introducing non-determinism while the second can be used to force big models [Baader and Hollunder, 1991; Schmidt-Schauss and Smolka, 1991; Donini *et al.*, 1992, 1997; Buchheit *et al.*, 1993]. Notice that with a bound of polynomial complexity, almost no exploration is possible. In a sense, this is at odds with the intuitive notion of inference, which calls for considering all possible outcomes given the present information.

Interestingly, the technological developments that took place in the intervening years have moved the boundaries of "tractability" further away. During a discussion at the 1998 Description Logic Workshop, Franz Baader put forward a bold claim. Nowadays, he argued, with the advances in both computing power and theorem proving techniques, tractability means something very near to ExpTime. Of course, this completely changes the perspective. The claim is supported by the performance of new theorem provers, as standard test beds like those proposed in [Heuerding and Schwendimann, 1996; Balsiger and Heuerding, 1998] have become too easy and fall in disuse. They are no longer appropriate for measuring the performance of state of the art provers (see [Horrocks *et al.*, 2000a] for an overview).

In sufficiently expressive languages, like $\mathcal{ALC}$ and extensions, all reasoning tasks we introduced can be reduced to instance checking (see [Donini *et al.*, 1994]). Let $\mathcal{L}$ be a language containing the $\mathcal{C}$ and $\mathcal{U}$ constructors, then subsumption and consistency in $\mathcal{L}$ can be reduced to instance checking. If $\mathcal{L}$ contains the $\mathcal{B}$ constructor, then relation checking can be reduced to instance checking.

PROPOSITION 2.6. [Reductions] *Given a knowledge base* Σ*,* $C_1, C_2 \in \mathsf{CON}(\mathcal{L})$*,* $R \in \mathsf{ROL}(\mathcal{L})$ *and* $a, b \in \mathsf{IND}$*, the following equivalences hold*

i. $\Sigma \models C_1 \sqsubseteq C_2$ *iff* $\Sigma \models a\!:\!(\neg C_1 \sqcup C_2)$*, for some* $a \in \mathsf{IND} \backslash \mathsf{IND}(\Sigma \cup \{C_1, C_2\})$*.*

ii. $\Sigma \models C \doteq \bot$ *iff* $\Sigma \models a\!:\!\neg C$*, for some* $a \in \mathsf{IND} \backslash \mathsf{IND}(\Sigma \cup \{C\})$*.*

iii. $\Sigma \models \bot$ *iff* $\Sigma \models a\!:\!C$*, for some* $C \in \mathsf{CON} \backslash \mathsf{CON}(\Sigma)$*.*

iv. $\Sigma \models (a, b)\!:\!R$ *iff* $\Sigma \models a\!:\!\exists R.\{b\}$*.*

A more delicate reduction is related to the transformation of T-Box and A-Box reasoning into *pure reasoning*, i.e., inference from an empty knowledge base. This is similar to what in classical logic is known as a deduction theorem. For example, for first-order logic the following property holds.

THEOREM 2.7. [Deduction theorem for first-order logic] *Let* $\Sigma \cup \{\varphi, \psi\}$ *be a set of sentences of first-order logic, then*

$$\Sigma \cup \{\varphi\} \models \psi \text{ iff } \Sigma \models \varphi \rightarrow \psi.$$

If Σ is a finite set or if the logic is compact, then $\Sigma \models \varphi$ can be reduced to $\{\} \models \varphi'$ by iterating the application of the deduction theorem. But a property like the deduction theorem does not necessarily hold. Here is where the restriction to simple and acyclic definitions we discussed in Section 2.2.2 comes to help. Clearly, with such restricted definitions we can do the following.

– Let $\Sigma = \langle T \cup \{B \sqsubseteq C\}, A\rangle$ be a knowledge base and let $B' \notin \mathsf{CON}(\Sigma)$, then

$$\langle T \cup \{B \sqsubseteq C\}, A\rangle \models a\!:\!D \text{ iff } \langle T \cup \{B \doteq C \sqcap B'\}, A\rangle \models a\!:\!D.$$

This is known as *completion* of definitions.

– Let $\Sigma = \langle T \cup \{B \doteq C\}, A\rangle$ be a knowledge base, then

$$\langle T \cup \{B \doteq C\}, A\rangle \models a\!:\!D \text{ iff } \langle T[B/C], A[B/C]\rangle \models a\!:\!D[B/C].$$

This is known as *unfolding* of definitions.

By repeatedly applying completion and unfolding of definitions we can transform $\Sigma \models \varphi$ into an equivalent task $\Sigma' \models \varphi'$ where Σ' has an empty T-Box, provided that definitions in Σ are simple and acyclic [Nebel, 1990b]. Notice, though, that the sizes of Σ' and φ' can be exponential with respect to the original Σ and φ. In certain languages, unfolding can be done "on the fly" avoiding this complexity explosion (see [Baader *et al.*, 1994; Lutz, 1999a, 1999b]). It is also straightforward to see that we have a "deduction theorem" for A-Boxes

$$\langle \{\}, A \cup \{b\!:\!B\}\rangle \models c\!:\!C \text{ iff } \langle \{\}, A\rangle \models \neg(b\!:\!B) \sqcup c\!:\!C. \tag{2.1}$$

And similarly for assertions $(a, b) : R$. But notice that standard description languages do not allow the Boolean combination of assertions as we did in (2.1). As before, we run into problems with some of the syntactical restrictions imposed by DLs. We will analyze the matter in detail in Chapter 4.

2.4 Constraint Systems

Even though we have formally discussed the different reasoning tasks, up to now we have said nothing about algorithms to decide any of them. Given that we will focus on languages extending $\mathcal{ALC}$, and the properties we listed in Proposition 2.6, we only need an algorithm to decide instance checking.

Calculi based on *constraint systems* were introduced by Schmidt-Schauss and Smolka in [1991] to decide satisfiability for empty knowledge bases in $\mathcal{ALC}$ and its sublanguages. Later, the framework was extended to decide different reasoning tasks for a variety of languages [Donini *et al.*, 1991, 1992, 1997; Baader and Hollunder, 1991]. Constraint systems are generalizations of tableau calculi [Ladner, 1977; Fitting, 1983; Rautenberg, 1983; Goré, 1999]. The constraint system we present below is from [Schaerf, 1994].

We assume without loss of generality that roles and concepts are given in negation normal form (i.e., negation is applied only to atomic concepts or to expressions of the form $\{a_1, \ldots, a_n\}$) and, furthermore, that the $^{-1}$ operator is only applied to atomic roles. Let VAR be a countably infinite set disjoint from IND, a *constraint* is a formula of one of the following forms

$$s\!:\!C \mid (s,t)\!:\!R \mid s \neq t \mid \forall x.x\!:\!C,$$

where $s, t \in \mathsf{IND} \cup \mathsf{VAR}$, $C \in \mathsf{CON}(\mathcal{L})$ and $R \in \mathsf{ROL}(\mathcal{L})$.

Let $\mathcal{I}$ be an interpretation, an $\mathcal{I}$*-assignment* is a function α that maps every variable in VAR to an element of $\Delta^{\mathcal{I}}$ and every individual a in IND to $a^{\mathcal{I}}$. We use pairs $\langle \mathcal{I}, \alpha\rangle$ to define *satisfiability of constraints*. Let $s^{\langle \mathcal{I},\alpha\rangle}$ be $s^{\mathcal{I}}$ if $s \in \mathsf{IND}$ and $\alpha(s)$ if $s \in \mathsf{VAR}$,

- $\langle \mathcal{I}, \alpha\rangle \models s\!:\!C$ iff $s^{\langle \mathcal{I},\alpha\rangle} \in C^{\mathcal{I}}$,
- $\langle \mathcal{I}, \alpha\rangle \models (s,t)\!:\!R$ iff $(s^{\langle \mathcal{I},\alpha\rangle}, t^{\langle \mathcal{I},\alpha\rangle}) \in R^{\mathcal{I}}$,
- $\langle \mathcal{I}, \alpha\rangle \models s \neq t$ iff $s^{\langle \mathcal{I},\alpha\rangle} \neq t^{\langle \mathcal{I},\alpha\rangle}$,
- $\langle \mathcal{I}, \alpha\rangle \models \forall x.x : C$ iff $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$.

A constraint φ is *satisfiable* if there is an interpretation $\mathcal{I}$ and an $\mathcal{I}$-assignment α such that $\langle \mathcal{I}, \alpha\rangle \models \varphi$. A *constraint system* $\mathcal{S}$ is a finite, non-empty set of constraints. A pair $\langle \mathcal{I}, \alpha\rangle$ *satisfies* $\mathcal{S}$ if $\langle \mathcal{I}, \alpha\rangle$ satisfies every constraint in $\mathcal{S}$, in which case we say that $\mathcal{S}$ is *satisfiable*. A knowledge base $\Sigma = \langle T, A\rangle$ in a language with the $\mathcal{C}$ and $\mathcal{U}$ constructors can be easily translated into a constraint system $\mathcal{S}_\Sigma$ by taking

$$\mathcal{S}_\Sigma = A \cup \{\forall x.x\!:\!\neg C \sqcup D \mid C \sqsubseteq D \in T\}.$$

The following proposition is immediate (notice the relation with Proposition 2.6).

Proposition 2.8. *Given a knowledge base* Σ*,* $C, D \in \mathsf{CON}(\mathcal{L})$*,* $R \in \mathsf{ROL}(\mathcal{L})$*,* $a, b \in \mathsf{IND}$*, and* $x \in \mathsf{VAR}$*, the following equivalences hold*

i. $\Sigma \models C \sqsubseteq D$ *iff* $\mathcal{S}_\Sigma \cup \{x : C \sqcap \neg D\}$ *is unsatisfiable.*
ii. $\Sigma \models C \doteq \bot$ *iff* $\mathcal{S}_\Sigma \cup \{x : C\}$ *is unsatisfiable.*
iii. $\Sigma \models \bot$ *iff* $\mathcal{S}_\Sigma$ *is unsatisfiable.*
iv. $\Sigma \models (a, b) : R$ *iff* $\mathcal{S}_\Sigma \cup \{a : \neg\exists R.\{b\}\}$ *is unsatisfiable.*
v. $\Sigma \models a : C$ *iff* $\mathcal{S}_\Sigma \cup \{a : \neg C\}$ *is unsatisfiable.*

In what follows, we will introduce a set of *completion rules* which, when applied to a constraint system $\mathcal{S}$, returns a constraint system $\mathcal{S}'$ such that $\mathcal{S}$ is satisfiable if and only if $\mathcal{S}'$ is. Furthermore, by repeatedly applying completion rules we will eventually either reach a *complete* constraint system where no further rules can be applied (implying that the system is satisfiable), or we will reach a clash (signaling that the system is contradictory). We first need some general definitions.

Definition 2.9. Let $\mathcal{S}$ be a constraint system, $x \in \mathsf{VAR}$, $s, t \in \mathsf{VAR} \cup \mathsf{IND}$, and $R \in \mathsf{ROL}(\mathcal{L})$. Then $\mathcal{S}[x/s]$ is the constraint system obtain by replacing each occurrence of x by s. We say that t is a *direct* R*-successor* of s in $\mathcal{S}$, if $(s, t) : R \in \mathcal{S}$, and we define *direct* R*-predecessors* similarly. Furthermore, t is a *direct successor* (*direct predecessor*) of s in $\mathcal{S}$ if it is a direct R-successor (direct R-predecessor) for some R. The *successor* (*predecessor*) relation is defined as the transitive closure of the direct successor (direct predecessor) relation. We say that s and t are *separated* in $\mathcal{S}$ if $s \not\doteq t \in \mathcal{S}$. Finally, we say that t *is a filler of* R *for* s *in* $\mathcal{S}$ if either t is an R-successor of s in $\mathcal{S}$, or one of the following conditions holds

- R is an atomic role and $(t, s) : R^{-1} \in \mathcal{S}$,
- R is of the form S^{-1} and $(t, s) : S \in \mathcal{S}$, or
- R is $R_1 \sqcap R_2$ and t is a filler of R_1 and R_2 for s in $\mathcal{S}$.

Figure 2.2 lists the different completion rules. The full set of rules handles the language $\mathcal{ALCNOBRI}$ with non-empty T- and A-boxes, and can, therefore, also handle sublanguages and simpler forms of knowledge bases.

What remains to do is to define the notion of a clash in a constraint system.

Definition 2.10. [Clash] We say that a constraint system $\mathcal{S}$ contains a *clash* if one of the following conditions holds.

i. For some s, $s : \bot \in \mathcal{S}$.
ii. For some s and some atomic concept A, $\{s : A, s : \neg A\} \subseteq \mathcal{S}$.
iii. For some s, $\{s : (\leq n\ R)\} \cup \{(s, t_i) : R \mid 1 \leq i \leq n + 1\} \cup \{t_i \not\doteq t_j \mid 1 \leq i < j \leq n + 1\} \subseteq \mathcal{S}$.
iv. For some s and $n > m$, $\{s : (\geq n\ R), s : (\leq m\ R)\} \subseteq \mathcal{S}$.
v. For some s, $s : \{a_1, \ldots, a_n\} \in \mathcal{S}$, and $s \neq a_j$ for all j.
vi. For some s, $s : \neg\{a_1, \ldots, a_n\} \in \mathcal{S}$, and $s = a_j$ for some j.

In [Schaerf, 1994], the following completeness result is stated.

– $\mathcal{S} \rightarrow_{\sqcap} \{s\!:\!C_1, s\!:\!C_2\} \cup \mathcal{S}$
  if $s\!:\!C_1 \sqcap C_2$ is in $\mathcal{S}$, and either $s\!:\!C_1$ or $s\!:\!C_2$ is not in $\mathcal{S}$;
– $\mathcal{S} \rightarrow_{\sqcup} \{s\!:\!D\} \cup \mathcal{S}$
  if $s\!:\!C_1 \sqcup C_2$ is in $\mathcal{S}$, neither $s\!:\!C_1$ nor $s\!:\!C_2$ is in $\mathcal{S}$, and $D = C_1$ or $D = C_2$;
– $\mathcal{S} \rightarrow_{\exists} \{(s,y)\!:\!R, y\!:\!C\} \cup \mathcal{S}$
  if $s\!:\!\exists R.C$ is in $\mathcal{S}$, there is no t such that t is a direct R-successor of s in $\mathcal{S}$ and $t\!:\!C$ is in $\mathcal{S}$, and y is a new variable;
– $\mathcal{S} \rightarrow_{\forall} \{t\!:\!C\} \cup \mathcal{S}$
  if $s\!:\!\forall R.C$ is in $\mathcal{S}$, t is a filler of R for s, and $t\!:\!C$ is not in $\mathcal{S}$;
– $\mathcal{S} \rightarrow_{\geq} \{(s,y_1)\!:\!R, \ldots, (s,y_n)\!:\!R\} \cup \{y_i \not\doteq y_j \mid 1 \leq i < j \leq n\} \cup \mathcal{S}$
  if $s\!:\!(\geq n\ R)$ is in $\mathcal{S}$, there do not exist n pairwise separated fillers of R for s in $\mathcal{S}$, and $y_1, \ldots, y_n$ are new variables;
– $\mathcal{S} \rightarrow_{\leq} \mathcal{S}[t/z]$
  if $s\!:\!(\leq n\ R)$ is in $\mathcal{S}$, s has more than n fillers of R for s and t, z are two fillers of R for s that are not separated;
– $\mathcal{S} \rightarrow_{\mathcal{O}} \mathcal{S}[x/a_i]$
  if $x\!:\!\{a_1, \ldots, a_n\}$ is in $\mathcal{S}$ and $1 \leq i \leq n$;
– $\mathcal{S} \rightarrow_{\mathcal{B}} \{(s,a)\!:\!R\} \cup \mathcal{S}$
  if $s\!:\!\exists R.\{a\}$ is in $\mathcal{S}$, and $(s,a)\!:\!R$ is not in $\mathcal{S}$;
– $\mathcal{S} \rightarrow_{\mathcal{R}} \{(s,t)\!:\!R_1, (s,t)\!:\!R_2\} \cup \mathcal{S}$
  if $(s,t)\!:\!R_1 \sqcap R_2$ is in $\mathcal{S}$, and either $(s,t)\!:\!R_1$ or $(s,t)\!:\!R_2$ is not in $\mathcal{S}$;
– $\mathcal{S} \rightarrow_{-1} \{(t,s)\!:\!R\} \cup \mathcal{S}$
  if $(s,t)\!:\!R^{-1}$ is in $\mathcal{S}$, and $(t,s)\!:\!R$ is not in $\mathcal{S}$;
– $\mathcal{S} \rightarrow_{\forall x} \{s\!:\!C\} \cup \mathcal{S}$
  if $\forall x.x\!:\!C$ is in $\mathcal{S}$, s appears in $\mathcal{S}$, and $s\!:\!C$ is not in $\mathcal{S}$.

Figure 2.2: Completion rules

THEOREM 2.11. *Let $\mathcal{S}$ be a constraint system, then it contains a clash only if $\mathcal{S}$ is not satisfiable, and a complete constraint system $\mathcal{S}$ is satisfiable if it contains no clash.*

We can prove that instance checking (and hence all other reasoning tasks we have introduced) for the full language $\mathcal{ALCNOBRI}$ is decidable (actually solvable in NEXPTIME). We will reduce it to satisfiability in C^2 (recall our discussion in Section 1.2.2) by providing a pair of mutually recursive translation functions ST_x and ST_y as we did in Definition 1.19, and following ideas from [Borgida, 1996].

PROPOSITION 2.12. *The instance checking problem for $\mathcal{ALCNOBRI}$ is decidable, as it can be reduced to satisfiability in C^2.*

PROOF. We define ST_x, ST_y being identical but swapping the positions of x and y. We first translate complex roles as follows. Remember that we assumed that the inverse operator $^{-1}$ was applied only to atomic roles. We consider each atomic role R in ROL as a binary predicate symbol in the first-order language.

$$\begin{array}{rcl} ST_x(R) & = & R(x,y), \text{ for } R \text{ an atomic role} \\ ST_x(R^{-1}) & = & R(y,x), \text{ for } R \text{ an atomic role} \\ ST_x(R_1 \sqcap R_2) & = & ST_x(R_1) \wedge ST_x(R_2). \end{array}$$

We can now provide the translation of complex concepts. Many constructs can be simulated in $\mathcal{ALCNOBRI}$ and we do not need to translate them. We consider each atomic concept C in CON as a unary predicate symbol.

$$\begin{array}{rcl} ST_x(C) & = & C(x), \text{ for } C \text{ an atomic concept} \\ ST_x(\neg C) & = & \neg ST_x(C) \\ ST_x(C_1 \sqcap C_2) & = & ST_x(C_1) \wedge ST_x(C_2) \\ ST_x(\exists R.C) & = & \exists y.(ST_x(R) \wedge ST_y(C)) \\ ST_x(\geq n\, R) & = & \exists^{\geq n} y.ST_x(R) \\ ST_x(\{a_1,\ldots,a_n\}) & = & \bigvee_{1\leq i\leq n}(x = a_i). \end{array}$$

Finally we translate terminological axioms and assertions ($(a,b){:}R$ can be simulated as $a{:}\exists R.\{b\}$).

$$\begin{array}{rcl} ST_x(C \sqsubseteq D) & = & \forall y.(ST_y(C) \rightarrow ST_y(D)) \\ ST_x(a{:}C) & = & ST_x(C)[x/a]. \end{array}$$

It is easy to prove that for a knowledge base $\Sigma = \langle T, A\rangle$ and an assertion $a{:}C$,

$$\Sigma \models a{:}C \text{ iff } \models \bigwedge_{\varphi\in T\cup A} (ST_x(\varphi)) \rightarrow ST_x(a{:}C).$$

As the formula on the right hand side is in C^2 we are done. QED

The reduction above also makes explicit how DLs are fragments of first-order logic. Another interesting fact that comes out from the reduction is that reasoning tasks in description logics are recast as validity of *sentences*. Or in other words, the *local* evaluation which is traditional in modal languages (satisfiability of a *formula with a free variable* at a given point in the model) has not received attention in the DL community. This is another point we will investigate in Chapter 4.

Going back to constraint systems, we should point out that the completion rules in Figure 2.2 do not constitute a decision method, as an infinite sequence of applications can arise (for example, the trivial interaction between the $\rightarrow_{\leq}$ and $\rightarrow_{\geq}$ rules where a new variable is generated and immediately identified with an existing one).

But the rules of the calculus can be carefully specialized, even to the point of obtaining not only decidability but also very sharp complexity results. For example, replacing the $\rightarrow_{\exists}$ and $\rightarrow_{\geq}$ rules by *trace versions* $\rightarrow_{T\exists}$ and $\rightarrow_{T\geq}$:

- $\mathcal{S} \rightarrow_{T\exists} \{(s,y){:}R, y{:}C\} \cup \mathcal{S}$
  if $s{:}\exists R.C$ is in $\mathcal{S}$, there is no t such that t is an R-successor of s in $\mathcal{S}$ and $t{:}C$ is in $\mathcal{S}$, y is a new variable, and for all constraint $(t,x){:}R$ in $\mathcal{S}$, t is a predecessor of s or $s = t$;

– $\mathcal{S} \rightarrow_{T\geq} \{(s,y_1):R,\ldots,(s,y_n):R\} \cup \{y_i \not\doteq y_j \mid 1 \leq i < j \leq n\} \cup \mathcal{S}$
if $s:(\geq n\ R)$ is in $\mathcal{S}$, there do not exist n pairwise separated fillers of R for s in $\mathcal{S}$, $y_1,\ldots,y_n$ are new variables, and for all constraint $(t,x):R$ in $\mathcal{S}$, t is a predecessor of s or $s = t$;

the decidability of concept satisfiability and subsumption for empty knowledge bases in $\mathcal{ALCNR}$ can be proved [Donini *et al.*, 1997]. In a similar way, the complexity of the different reasoning tasks for sublanguages of $\mathcal{ALCO}$ has been carefully mapped out (see [Schaerf, 1994]). In [Tobies, 2000b] the constraint system approach is used to prove that pure concept satisfiability, i.e., with respect to empty knowledge bases, in $\mathcal{ALCQR}$ is PSPACE-complete. The constructor $\mathcal{Q}$ of qualified number restrictions is an extension of $\mathcal{N}$ and we will discuss these results further in Section 4.5.5. And in [Horrocks *et al.*, 1999, 2000b, 2000c; Tobies, 2000a] the decidability of both T-Box and A-Box reasoning is proved for very expressive description logics together with a complexity analysis of sublanguages, also by means of constraint systems.

But perhaps the most important characteristic of constraint systems, from a computational logic perspective, is that *they readily provide implementations*. Implementing the completion rules of Figure 2.2 is simple, and many DL theorem provers, like for example FACT [Horrocks, 1999], DLP [Patel-Schneider, 1998] or RACE [Haarslev and Möller, 1999], are based on this technique. These systems have been highly optimized and are able to cope with extensive knowledge bases in very expressive description languages.