14

HYBRID LOGICS

Carlos Areces and Balder ten Cate

1 What are Hybrid Logics? . . . 822
  1.1 Basic Definitions . . . 824
2 History . . . 826
  2.1 The Foundational Work of Prior . . . 826
  2.2 The Sofia School . . . 828
  2.3 Very Expressive Hybrid Languages . . . 831
3 Model Theory . . . 832
  3.1 Completeness . . . 832
  3.2 Expressive Power and Characterization . . . 837
  3.3 Interpolation and Beth Definability . . . 844
4 Decidability and Complexity . . . 846
5 Proof Theory . . . 852
  5.1 Sequent Calculus . . . 852
  5.2 Natural Deduction Calculus . . . 854
  5.3 Tableau Calculus . . . 855
  5.4 Resolution Calculus . . . 857
6 Relation with Other Fields . . . 858
7 Discussion . . . 863

This chapter provides a modern overview of the field of hybrid logic. Hybrid logics are extensions of standard modal logics, involving symbols that name individual states in models. The first results that are nowadays considered as part of the field date back to the early work of Arthur Prior in 1951. Since then, hybrid logic has gone through a number of revivals and reinventions. Nowaways, it is a field of research in its own right, with a wealth of results, techniques, and applications.

Our main aim, in this chapter, is to provide a coherent picture of the current state of affairs in the field of hybrid logic. Rather than a comprehensive summary, we will try to give the reader a taste for the type of results and techniques that we consider hallmarks of the field. In some cases, we will only state results, with pointers to relevant literature, while in other cases we will provide full proofs.

In Section 1, we give an intuitive introduction to hybrid logics, with examples of the extra expressive power offered by the hybrid operators. This section also contains the basic definitions of syntax and semantics that are used throughout the chapter. In Section 2 we provide a short history of the field, discussing the work of Prior in the 50s, of the Sofia School in the 80s, and the work on very expressive hybrid languages in the 90s. Sections 3 and 4 form the core of

the chapter. They contain the most important techniques and results in the field, with respect to completeness, expressive power, frame definability, interpolation and complexity. In Section 5 we briefly present proof systems for hybrid languages (sequents, natural deduction, tableaux, and resolution), and we discuss some issues concerning the development of automated provers based on them. In Section 6 we comment on connections with related areas (some of which are discussed in detail in other chapters of this handbook). Section 7 finishes the chapter with a summary and general perspectives.

1 WHAT ARE HYBRID LOGICS?

In their simplest form, hybrid languages are modal languages that have special symbols to name individual states in models. These new symbols, which are called *nominals*, enter the stage gracefully: we simply add a new sort of atomic symbols $\mathsf{NOM} = \{i, j, k, \ldots\}$ disjoint from the set PROP of propositional variables and let them combine freely in formulas. For example, if i is a nominal and p and q are propositional variables, then

$$\Diamond(i \wedge p) \wedge \Diamond(i \wedge q) \rightarrow \Diamond(p \wedge q), \tag{1}$$

is a well formed formula. Now for the important twist: since nominals name individual states in the model, they denote *singleton sets*. In other words, they are true at a unique point in the model. Once this step has been taken, the whole landscape changes. For example, (1) becomes a validity: let $\mathcal{M}$ be a model, m a state in the domain of $\mathcal{M}$, and suppose $\mathcal{M}, m \models \Diamond(i \wedge p) \wedge \Diamond(i \wedge q)$. Then some successor state m' of m satisfies $i \wedge p$, and some successor state m'' of m satisfies $i \wedge q$. Since i is a nominal, it is true at a unique point in $\mathcal{M}$. Hence $m' = m''$ and we have $\mathcal{M}, m \models \Diamond(p \wedge q)$. Note that (1) could be falsified if i were an ordinary propositional variable.

When we realize the potential that nominals have, an interesting idea suggests itself: to introduce, for each nominal i, an operator $@_i$ that allows us to jump to the point named by i. The formula $@_i\varphi$ (read "at i, φ") moves the point of evaluation to the state named by i and evaluates φ there. These operators satisfy many nice logical properties. For a start, each $@_i$ is a normal modal operator: it satisfies the distributivity axiom ($@_i(\varphi \rightarrow \psi) \rightarrow (@_i\varphi \rightarrow @_i\psi)$) and the necessitation rule (if φ is valid, then $@_i\varphi$ is also valid). Moreover, it is self-dual: $@_i\varphi$ is equivalent to $\neg@_i\neg\varphi$. In an intuitive sense, the $@_i$ operators provide a bridge between semantics and syntax by internalizing the satisfaction relation '$\models$' into the logical language:

$$\mathcal{M}, w \models \varphi \text{ iff } \mathcal{M} \models @_i\varphi, \text{ where } i \text{ is a nominal naming } w.$$

For this reason, these operators are usually called *satisfaction operators*.

Aiming to make full use of the flexibility provided by direct reference to specific points in the model naturally leads to further enrichment of the language. One possibility would be to have not only names for individual states but also variables ranging over states, with corresponding quantifiers. We would then be able to write formulas like

$$\forall y.\Diamond y. \tag{2}$$

The first-order translation of this formula is $\forall y.\exists z.(R(x, z) \wedge z = y)$ or, simply, $\forall y.R(x, y)$, forcing the current state to be related to all states in the domain. The $\forall$ quantifier is very expressive. As discussed in [32], even the basic modal language extended with state variables and this universal quantifier is undecidable. Moreover, $\forall$ and $@$ together give us already full first-order

expressive power (cf. Section 3.2). Nevertheless, the $\forall$ quantifier is historically important. The earliest treatments are probably those of [117, 118, 46].

The $\forall$ quantifier is very "classical." If we think modally, and remember that evaluation of modal formulas takes place *at a given point*, a different kind of binder suggests itself. The $\downarrow$ binder binds variables to points but, unlike $\forall$, it binds to the *current* point. In essence, it enables us to create a name for the here-and-now, and refer to it later in the formula. For example, the formula

$$\downarrow y.\Diamond y \tag{3}$$

is true at a state m iff m is related to itself. The intuitive reading of (3) is quite straightforward: the formula says "call the current state y and check that y is reachable." The difference between $\forall$ and $\downarrow$ is subtle, but important. $\forall$ is *global*, in the sense that formulas containing $\forall$ are not preserved under generated submodels [32]. On the other hand, $\downarrow$ is intrinsically local and, as we will show in Theorems 16 and 18, it can be characterized in terms of the operation of taking generated submodels.

Like $\forall$, the $\downarrow$ binder has been invented independently on several occasions. For example, in [122], $\downarrow$ is introduced as part of an investigation into temporal semantics and temporal databases, [131] uses it to aid reasoning about automata, it is related to the freeze operator in [88], and [52] employs it as part of his treatment of indexicality. However, none of the systems just mentioned allows the free syntactic interplay of variables with the underlying propositional logic; that is, they make use of $\downarrow$, but in languages that are not fully hybrid. The earliest paper to introduce it into a fully hybrid language seems to be [77].

Note that satisfaction operators work in perfect coordination with $\downarrow$. Whereas $\downarrow$ "stores" the current point of evaluation (by binding a variable to it), the satisfaction operators enable us to "retrieve" stored information by shifting the point of evaluation in the model. By using the "storing and retrieving" intuition it is easy to define complex properties. For example, Kamp's temporal *until* operator U (with semantics: $U(\varphi, \psi)$ is true at a state m if there is a future state m' where φ holds, such that ψ holds in all states between m and m') can be defined as follows:

$$U(\varphi, \psi) := \downarrow x.\Diamond\downarrow y.(\varphi \wedge @_x\Box(\Diamond y \rightarrow \psi)).$$

Let us see how this work. First, we name the current state x using $\downarrow$, and use the $\Diamond$ operator to find a suitable successor state, which we call y, where φ holds. Without the $@$ operator we would be stuck in that successor state, but we can use $@$ to go back to x and demand that in all successors of x having y as a successor, ψ holds.

Summarizing the above discussion, we can say that the term *hybrid logic* refers to a family of extensions of the basic hybrid language with devices that, in one way or another, allow for explicit reference to individual states of the Kripke model. But, why are hybrid logics called *hybrid*?

One explanation comes from the work of Arthur Prior in the 1950s. As we will discuss more in detail in Section 2, Prior was interested in the relation between what McTaggart called the A-series and B-series of time [109]. Following McTaggart's analysis of time in terms of the A-series of past, present and future and the B-series of earlier and later, Prior discusses two logical systems: the I-calculus aims to capture the properties of the B-series and takes variables ranging over instants as primitive, while the T-calculus examines tenses and takes variables ranging over propositions. In [117, Chapter V.6], Prior proposes a way to develop the I-calculus *inside* the T-calculus, and for this he allows instant-variables to be used together with propositional variables. He will call this step "the third grade of tense-logical involvement" in [118, Chapter XI], where instant-variables are treated as representing (special) propositions. From this perspective, the