14
Formula Evaluation

Logical languages usually have a model-theoretic semantics defining when a formula φ is true in a model $\boldsymbol{M}$, perhaps with an auxiliary setting. The paradigm is first-order logic, with its notion $\boldsymbol{M}, s \models \varphi$ where s is an assignment of objects in $\boldsymbol{M}$ to variables. Now, stepwise evaluation of first-order assertions can be cast dynamically as a game of evaluation for two players. "Verifier" claims that φ is true in the setting $\boldsymbol{M}, s$, "falsifier" that it is false. This is our most basic logic game. In this chapter we explain first-order evaluation games, establish their adequacy with respect to truth and falsity, explore their more general game-theoretic character, demonstrate how other logics can be gamified in the same style, and identify some general issues of game logic behind first-order games, including the role of players' strategies and game operations.[169]

14.1 Evaluation games for predicate logic

Two parties disagree about a proposition φ in some situation $\boldsymbol{M}, s$: *verifier* $\boldsymbol{V}$ claims that it is true, *falsifier* $\boldsymbol{F}$ that it is false. Here are the natural moves of defense and attack in the first-order evaluation game, that we will indicate henceforth as $\boldsymbol{game}(\varphi, \boldsymbol{M}, s)$.

Definition 14.1 Moves in evaluation games

The moves of *evaluation games* follow the inductive construction of formulas. They

169 Games like this occur in Hintikka (1973). Since then, evaluation games have been given for many logics. Hintikka & Sandu (1997) has a game-theoretical semantics for natural language, and Chapter 21 will pursue the resulting independence-friendly logic.

involve typical notions in the dynamics of games, such as choice, switch, and continuation, in dual pairs with both players allowed the initiative once:

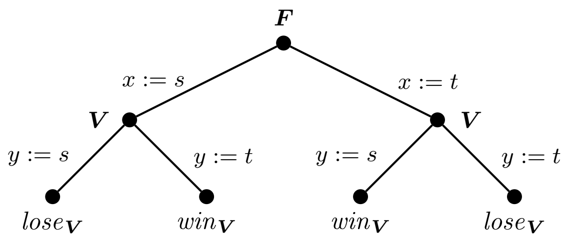
atoms $Pd, Rde, \ldots$	$\boldsymbol{V}$ wins if the atom is true, $\boldsymbol{F}$ if it is false
disjunction $\varphi \vee \psi$	$\boldsymbol{V}$ chooses which disjunct to play
conjunction $\varphi \wedge \psi$	$\boldsymbol{F}$ chooses which conjunct to play
negation $\neg\varphi$	role switch between the two players, play continues with respect to φ

Next, the quantifiers make players look inside $\boldsymbol{M}$'s domain of objects:

existential $\exists x\varphi(x)$	$\boldsymbol{V}$ picks an object d, play continues with $\varphi(d)$
universal $\forall x\varphi(x)$	the same move, but now for $\boldsymbol{F}$

Here the clause for atoms may look circular, but one might think of it as the players consulting the model to see whether it supports such a bottom-level statement. As for complex structure, the schedule of the game is determined by the form of the statement φ. ■

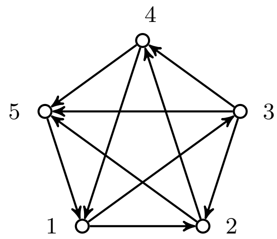
Example 14.1 Formulas and schedule of play
To see how this works, consider a model $\boldsymbol{M}$ with two objects s, t. Here is a game for $\forall x \exists y\, x{\neq}y$, pictured as a tree of moves, with the scheduling from top to bottom:



We interpret this as a game of perfect information: players know throughout what has happened. Falsifier starts, and verifier must respond. There are four possible plays, with two wins for each player. But verifier has a winning strategy, in the standard sense of our earlier chapters. ■

Trees such as this are not a complete definition of the game yet, but for many purposes, we are better off without further detail. Evaluation games for slightly more complex formulas in richer models have proved attractive in teaching logic.

EXAMPLE 14.2 Find noncommunicators
Consider the following communication network with arrows for directed links, and with all self-loops present but suppressed in the drawing:



The formula $\forall x \forall y(Rxy \vee \exists z(Rxz \wedge Rzy))$ says that every two nodes in this network can communicate in at most two steps. Here is a run of the evaluation game:

player	*move*	*next formula*
$\boldsymbol{F}$	picks 2	$\forall y(R\boldsymbol{2}y \vee \exists z(R\boldsymbol{2}z \wedge Rzy))$
$\boldsymbol{F}$	picks 1	$R\boldsymbol{2}\,\boldsymbol{1} \vee \exists z(R\boldsymbol{2}z \wedge Rz\boldsymbol{1})$
$\boldsymbol{V}$	chooses	$\exists z(R\boldsymbol{2}z \wedge Rz\boldsymbol{1})$
$\boldsymbol{V}$	picks 4	$R\boldsymbol{2}\,\boldsymbol{4} \wedge R\boldsymbol{4}\,\boldsymbol{1}$
$\boldsymbol{F}$	chooses	$R\boldsymbol{4}\,\boldsymbol{1}$
test	$\boldsymbol{F}$ loses	

Falsifier started with a threat by picking object 2, but then picked 1. Verifier chose the true right conjunct, and picked the witness 4. Now, falsifier loses with either choice. Still, falsifier could have won, by choosing object 3 that 2 cannot reach in $\leq$2 steps. Falsifier even has another winning strategy, namely, x=5, y=4. ∎

In this way, each formula φ is a game form of fixed depth but indefinite branching width, with a schedule of turns and moves. It becomes a real game when a model $\boldsymbol{M}$ is given that supplies possible quantifier moves and outcomes for atomic tests, while an assignment s to the free variables in φ sets the initial position of the game.

14.2 Truth and winning strategies of verifier

In our first example, participants were not evenly matched. Player $\boldsymbol{V}$ can always win: after all, a verifier is in line with the truth of the matter. More precisely, $\boldsymbol{V}$ has a *winning strategy*, a map from $\boldsymbol{V}$'s turns to moves following which guarantees, against any play by $\boldsymbol{F}$, that the game ends in outcomes where $\boldsymbol{V}$ wins. $\boldsymbol{F}$ has no winning strategy, as this would contradict $\boldsymbol{V}$'s having one.[170] Even more can be said. $\boldsymbol{F}$ does not have a losing strategy either: $\boldsymbol{F}$ cannot force $\boldsymbol{V}$ to win, but in our example, player $\boldsymbol{V}$ does have a losing strategy. Thus, players' powers of controlling outcomes in a game may be quite different.

Here is the key to the behavior of evaluation games, the "success lemma."

FACT 14.1 The following are equivalent for all models $\boldsymbol{M}, s$ and formulas φ:

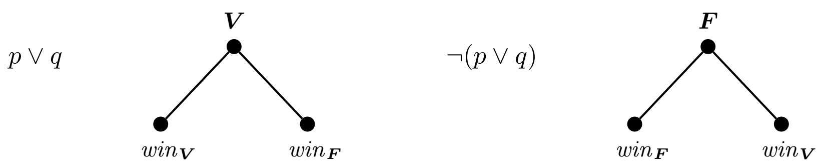
(a) $\boldsymbol{M}, s \models \varphi$, (b) $\boldsymbol{V}$ has a winning strategy in $\boldsymbol{game}(\varphi, \boldsymbol{M}, s)$.

Proof The proof is a direct induction on formulas. One shows simultaneously:

If a formula φ is true in $(\boldsymbol{M}, s)$, then verifier has a winning strategy.
If a formula φ is false in $(\boldsymbol{M}, s)$, then falsifier has a winning strategy.

The steps show the close analogy between logical operators and ways of combining strategies.[171] The following typical cases will give the idea. (a) If $\varphi \vee \psi$ is true, then at least one of φ or ψ is true, say, φ. By the inductive hypothesis, $\boldsymbol{V}$ has a winning strategy σ for φ. But then $\boldsymbol{V}$ has a winning strategy for the game $\varphi \vee \psi$: the first move is *left*, after which the rest is the strategy σ. (b) If $\varphi \vee \psi$ is false, both φ and ψ are false, and so by the inductive hypothesis, $\boldsymbol{F}$ has winning strategies σ and τ for φ and ψ, respectively. But then the combination of an initial wait-and-see step plus these two is a winning strategy for $\boldsymbol{F}$ in the game $\varphi \vee \psi$. If $\boldsymbol{V}$ goes left in the first move, then $\boldsymbol{F}$ should play σ, while, if $\boldsymbol{V}$ goes right, $\boldsymbol{F}$ should play strategy τ. (c) If the formula φ is a negation $\neg\psi$ we use a role switch.

EXAMPLE 14.3 Role Switch
Consider the game for a formula $p \vee q$ in a model where p is true and q is false, as well as its dual game $\neg(p \vee q)$, that switches all turns and win markings:



170 Playing two winning strategies against each other yields a contradiction at the end.

171 This inductive proof is virtually the argument for Zermelo's Theorem in Chapter 1.

The second game works out to that for the De Morgan equivalent $\neg p \wedge \neg q$. ■

Thus, strategies for $\boldsymbol{V}$ in a game for $\neg\psi$ are strategies for $\boldsymbol{F}$ in the game for ψ, and vice versa. Now we prove case (c). Suppose that $\neg\psi$ is true. Then ψ is false, and by the inductive hypothesis, $\boldsymbol{F}$ has a winning strategy in the ψ-game forcing an outcome in the set of $\boldsymbol{F}$'s winning positions. But this is a strategy for $\boldsymbol{V}$ in the $\neg\psi$-game, and indeed one forcing a set of winning positions for $\boldsymbol{V}$. The other direction is similar. ■

This is our first link between a key notion in logic (truth) and one in game theory (strategy). We will broaden the interface as we continue. Some critics see the success lemma as showing how games yield nothing new. To them, a game-theoretic analysis is good only if it captures some pre-existing logical notion. Our focus is the opposite: what new themes are intrinsic to games, and might enrich the old agenda of logic?

14.3 Exploring the game view of predicate logic

Simple as it is, there is more to the success lemma than meets the eye. In particular, this result suggests new perspectives on what makes standard predicate logic tick. Many technical distinctions to be formulated in the following discussion will recur in subsequent chapters.

Different winning strategies Truth occurs if and only if there is a winning strategy for player $\boldsymbol{V}$, and likewise for falsity and $\boldsymbol{F}$. But there can be more than one such strategy. For instance, $\boldsymbol{F}$ had two winning strategies in our Example 14.2, using two different counterexamples to the claim. Thus, winning strategies are more refined semantic objects than standard truth values, that we might call reasons for truth or falsity.

Games and game boards The success lemma compares two semantic settings. One is the model $\boldsymbol{M}$, or its associated space of assignments s of individual objects $s(x)$ to all relevant first-order variables x. Here a notion from Chapter 11 returns. This space serves as a "game board," a setting where evaluation games can be played, or even other games. Compare a Chess board with possible positions. Chess expands this with conventions, defining turns for players, as well as their winning positions. The latter are game-internal: there is nothing intrinsic to the distribution of pieces on the board that makes it a win for White or Black.

Comparing two different languages The success lemma compares the game and its board using expressions from different languages appropriate to them:

$$\boldsymbol{V} \text{ has a winning strategy in } \boldsymbol{game}(\varphi, \boldsymbol{M}, s) \quad \text{iff} \quad \boldsymbol{M}, s \models \varphi$$

The expression on the left can be rewritten in a game language referring to forcing powers of players (cf. Chapter 11), while that on the right-hand side is best viewed as a modal formula referring to actions on the board, as in Chapter 1:

$$\boldsymbol{game}(\varphi, \boldsymbol{M}, s) \models win_{\boldsymbol{V}} \quad \text{iff} \quad \boldsymbol{M}, s \models \varphi$$

This dual perspective can be generalized. On the left, one can talk about both players' powers for forcing any set of positions in the game. This corresponds to nested substitutions in modal assertions about the game board on the right.

The general topic of matching games and game boards will be pursued in greater depth in Chapters 19 and 24.

Defining the games formally Defining complete trees for logic games is largely routine. Still, formalization brings out interesting twists to understanding first-order logic. Let us define the tree for $\boldsymbol{game}(\varphi, \boldsymbol{M}, s)$ as follows. Nodes are all pairs

$$(s, \psi) \qquad \text{where } s \text{ is an } \boldsymbol{M}\text{-assignment, and } \psi \text{ is a subformula of } \varphi$$

Game moves reflect the earlier ones, changing one or both components of a state. In particular, atomic tests do not change the state, while choices only change its formula, moving from a current node $(s, \varphi \vee \psi)$ to one of its daughters (s, φ) and (s, ψ). But formalizing the other rules leads to departures from received views in predicate-logical semantics. Consider assignment change with quantifiers. Starting at $(s, \exists x\psi)$, verifier chooses an object d from the domain of $\boldsymbol{M}$, and s is set to $s[x := d]$. Play then continues with $\psi(d)$: that is, it starts afresh from the formal game state $(s[x := d], \psi)$. But this analysis suggests that, unlike in standard logical syntax, we can view the quantifier symbol $\exists x$ by itself as a separate interpretable entity, and more specifically, that *quantifiers are atomic games* of object picking. Standard thinking assimilates quantifiers to Boolean disjunctions or conjunctions. By contrast, here, the real game operation involved in $\exists x\psi$ is *sequential composition*, gluing the game for ψ after the independent atomic game for $\exists x$. On this view,

> *Predicate-logical semantics is really a system of games of object picking and fact testing, related by suitable game operations.*

Next, we need game-internal predicates of turn taking and winning. A formula φ tells us who is to move at which stage, although we need to take care with role switches for negations.

DEFINITION 14.2 Formal game trees
We define $\boldsymbol{game}(\varphi, \boldsymbol{M}, s)$ inductively, for any assignment s, starting from an initial state (s, φ). The first two clauses are for the two kinds of atomic game:

(a) $\boldsymbol{game}(\varphi, \boldsymbol{M}, s)$ for atomic φ is a one-node end game, which is a win for verifier if $\boldsymbol{M}, s \models \varphi$, and for falsifier otherwise.

(b) $\boldsymbol{game}(\exists x, \boldsymbol{M}, s)$ is a one-move game starting at s where it is $\boldsymbol{V}$'s turn, with possible moves to any state $s[x := d]$, always ending in a win for $\boldsymbol{V}$.

Next we turn to game constructions:

(c) $\boldsymbol{game}(\varphi \vee \psi, \boldsymbol{M}, s)$ is the disjoint union of the two games $\boldsymbol{game}(\varphi, \boldsymbol{M}, s)$ and $\boldsymbol{game}(\psi, \boldsymbol{M}, s)$ put under a common root $(s, \varphi \vee \psi)$ that is $\boldsymbol{V}$'s turn.

(d) $\boldsymbol{game}(\varphi \wedge \psi, \boldsymbol{M}, s)$ is defined likewise, but with an initial turn for $\boldsymbol{F}$.

(e) $\boldsymbol{game}(\neg\varphi, \boldsymbol{M}, s)$ is $\boldsymbol{game}(\varphi, \boldsymbol{M}, s)$ with turn and win markings reversed.

The negation switch was illustrated with the success lemma. Finally, to deal with quantifiers, we add a clause for an operation of composition for evaluation games:

(f) $\boldsymbol{game}(\varphi\,;\psi, \boldsymbol{M}, s)$ is the tree arising by first taking $\boldsymbol{game}(\varphi, \boldsymbol{M}, s)$ and continuing at end states with assignment t with a copy of $\boldsymbol{game}(\psi, \boldsymbol{M}, t)$.

These tree constructions for games will return in Chapters 20 and 25. ■

Emancipation of syntax The above semantics interprets more than just the usual well-formed formulas. For instance, it makes perfect sense to play a game for the string

$$Px \;;\; \exists x$$

which translates to "First test that P holds for object $s(x)$ under the current assignment s, then change the value of x, and stop." In contrast with this, the game for $\exists x\,; Px$ would first change the value of $s(x)$ to obtain an object with the property P. With composition as a new syntax construction, predicate logic extends to a language of discourse chunks $\alpha\,;\,\beta\,;\,\ldots$ that might be interesting to axiomatize.[172] Even not-so-well-formed formulas now get their chance.

Propositions versus activities Finally, the reader should beware of a common confusion. Reading formulas φ in their ordinary notation as evaluation games makes them serve a double role: as a static *proposition* in truth-conditional semantics, or as a *game* in our new semantics. A game is not a statement, but a dynamic activity. To be sure, one can state propositions about games, say, that verifier has a winning strategy in the φ-game. But this statement is not that game itself, and it does not exhaust the latter's content. Much of the original literature on game semantics conflates these two readings of first-order syntax, resulting in the prejudice that existence statements about verifier's winning strategies are all there is to know to the game-theoretic meaning of φ.[173]

14.4 Game-theoretic aspects of predicate logic

Now we explore connections with game theory, a theme raised in the Introduction. First, via the success lemma, logical laws acquire game-theoretic import.

Determinacy Our first encounter between logic and game theory was the following simple but telling observation.

FACT 14.2 Excluded middle $A \vee \neg A$ expresses determinacy of evaluation games; that is, one of the two players must always have a winning strategy.

Evaluation games turned out to be determined because of Zermelo's Theorem that all zero-sum two-player games of finite depth are determined (cf. Chapter 1).

172 We also get new distinctions. Consider a "bounded quantifier" $\exists y(Rxy \wedge \varphi)$, inducing a game where verifier chooses an object d for y, then falsifier chooses a conjunct, and one either tests the atom $R^{\boldsymbol{M}}s(x)d$, or play continues at $(s[y := d], \varphi)$. It seems more natural to package this as one atomic game: verifier picks an R-successor of $s(x)$. This is won by verifier if she can produce such an object, and lost otherwise. In extended first-order syntax, this would be the game for $\exists y\,;\,Rxy$. This is the view that underlies our later evaluation games for modal logic, and it will return in the theoretical analysis of logic games as a format for game algebra in Chapter 24.

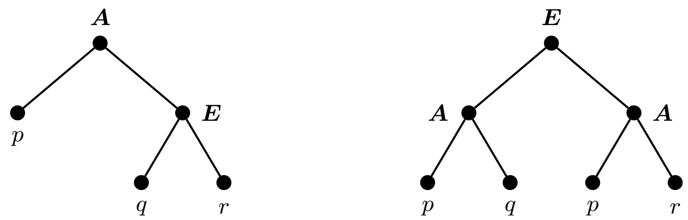
173 Significantly, two similar levels, of actions and propositions, were carefully kept separate in the syntax of propositional dynamic logic PDL (Chapters 1 and 4), with *programs* and *formulas*.

Infinite evaluation games will also be relevant later in this chapter, and there, we need to appeal to further results such as the Gale-Stewart Theorem of Chapter 5.

Game equivalence In the Introduction, we also saw an issue of game equivalence.

FACT 14.3 The propositional distribution law states that the evaluation games for its formulas are equivalent in terms of players' forcing powers.

EXAMPLE 14.4 Switching games with invariant powers
Switching between games $A \wedge (B \vee C)$ and $(A \wedge B) \vee (A \wedge C)$ transforms turns for players without affecting their strategic powers concerning outcomes:



The relevant computations were given in the Introduction and Chapter 1. ■

Chapter 11 contained an abstract perspective on powers that fits this equivalence precisely. In particular, Boolean operations turn into general operations on games. A matching algebra of power equivalence will be studied in Chapters 19 and 21.

Syntactic normal forms Propositional normal forms now serve as normal forms for games. The same is true for first-order quantificational prenex forms such as

$$\forall x \exists y \forall z \, Q(x, y, z)$$

with all quantifiers moved in front, dividing object picking moves into alternating blocks for each player, fixing scheduling without affecting players' powers. Also relevant are Skolem forms, taking first-order formulas to second-order equivalents

$$\exists f_1 \cdots \exists f_k \, \forall x_1 \cdots \forall x_m$$

followed by a quantifier-free propositional part, which will be used in Chapter 22.

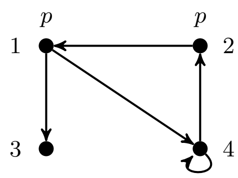
Here, an earlier caveat applies. Skolem forms are better read as statements about games, viz. the existence of strategies, rather than as games in themselves. Still, one can view second-order formulas such as $\exists x \exists g \forall x \forall y \, Q(x, f(x), y, g(x, y))$ as defining a new evaluation game, where verifier picks a strategy at the start, after which falsifier has a go, and finally a regular propositional game is played.[174]

14.5 Gamification: Variations and extensions

Evaluation games exist for many logical languages. The above explanations provide almost automatic gamifications, provided that the truth conditions employ quantifiers and connectives. For instance, the Skolem game mentioned at the end of the preceding section was an evaluation game for a second-order language, that goes as before, letting players now also choose functions for second-order quantifiers $\exists f$ (or, in other second-order games, sets), in addition to objects for first-order quantifiers $\exists x$. In this section, we discuss a few further illustrations with additional points.

Basic modal logic Consider the basic modal language over models $\boldsymbol{M} = (W, R, V)$ used in Parts I and II of this book. We start with a simple variation on first-order evaluation games, showing how they transfer to modal languages. Accessibility encodes moves that can be made to get from one world or state to another.

Example 14.5 A modal model
Consider the following graph with four worlds and accessibilities as indicated:



The formula $\Diamond\Box\Diamond p$ is true in states 1 and 4, but false in states 2 and 3. ■

Modal evaluation games search through such a model, with two key moves:

$\Diamond$	verifier chooses a successor of the current world
$\Box$	falsifier chooses a successor of the current world

174 Redescriptions are frequent with logic games. Switching between games raises general issues of game equivalence (cf. Chapter 1). We will discuss this theme in Chapter 18.

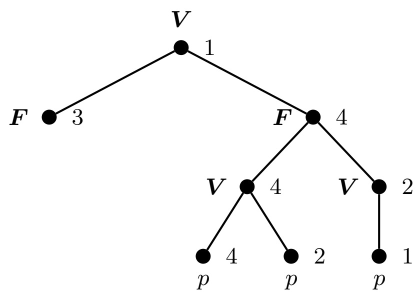
Game states are pairs (*state*, *formula*). Players lose when defending an atom that fails at the current state, or when they must choose a successor but cannot. This is like the bounded first-order quantifier $\exists y(Rxy \wedge Py)$ discussed earlier.

Again we have a modal success lemma:

$$\boldsymbol{M}, s \models \varphi \quad \text{iff} \quad \boldsymbol{V} \text{ has a winning strategy for the } \varphi\text{-game in } \boldsymbol{M} \text{ from } s.$$

EXAMPLE 14.6 A modal evaluation game

The graph of Example 14.5 induces the following game tree for $\Diamond\Box\Diamond p$ at state 1:



Here, $\boldsymbol{V}$ has two winning strategies: *left*, and *right* ; ⟨*right*, *down*⟩. These are the two ways of verifying $\Diamond\Box\Diamond p$ in the given model at world 1. The game also illustrates the well-known feature of losing when a player must move but cannot. ■

The same games work for polymodal languages with indexed operators $\langle a \rangle$.

Modal μ-calculus A more drastic change in evaluation games is needed for the fixed point logics in Parts I and II of this book. To define their games, we need to explain their central idea of recursion in a bit more detail than before, starting with an important special case. Recall the modal fixed point logic providing the recursive definition used to analyze Zermelo's algorithm in Chapter 1, which returned in many places, including Chapter 13 on solving games in strategic form. We present the basics here; interested readers may look to van Benthem (2010b) and, especially, Venema (2007) for more didactic detail.

DEFINITION 14.3 Modal μ-calculus

The modal *μ-calculus* extends basic modal logic with a syntactic operator $\mu p \bullet \varphi(p)$

in which all occurrences of p in φ are positive.[175] In any model $\boldsymbol{M}$, the new operator $\mu p \bullet \varphi(p)$ defines the *smallest fixed point* with respect to set inclusion for the following set operator $\varphi^{\boldsymbol{M}}$ on the model associated with the formula $\varphi(p)$:

$$\varphi^{\boldsymbol{M}}(X) = \{s \in \boldsymbol{M} \mid \boldsymbol{M}[p := X], s \models \varphi\}$$

Here a fixed point of a function F is an argument X such that $F(X) = X$. ■

By the positive syntactic occurrence of p in $\varphi(p)$, we can easily show the following important property.

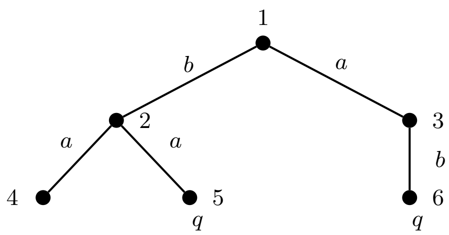
FACT 14.4 The map $\varphi^{\boldsymbol{M}}$ is monotonic for inclusion: $X \subseteq Y \Rightarrow \varphi^{\boldsymbol{M}}(X) \subseteq \varphi^{\boldsymbol{M}}(Y)$.

The definition relies on the following fact, called the Tarski-Knaster Theorem.

FACT 14.5 Every monotonic operator F on a power set has a smallest and a greatest fixed point in the inclusion order.[176]

A greatest fixed point operator $\nu p \bullet \varphi(p)$ is defined analogously. This powerful new formalism allows us to define properties of states in a model by recursion.

EXAMPLE 14.7 Fixed point evaluation
Consider the formula $\mu p \bullet (q \vee \langle a \rangle p)$ in the following model:



175 Positive occurrence means that, counting from the outside of φ, p lies in the scope of an even number of negations. Some formulas positive in p are $\neg q \vee \langle a \rangle p$, $\neg \langle a \rangle \neg p$, and $\neg \mu q \bullet (\neg \langle a \rangle \neg q \wedge \neg p)$.

176 The proof of the Tarski-Knaster Theorem shows how the smallest fixed point $F_* = \bigcap \{X \subseteq A \mid F(X) \subseteq X\}$, while the greatest fixed point $F^* = \bigcup \{X \subseteq A \mid X \subseteq F(X)\}$. One can also think of these as reached by approximation through the ordinals. F_* is the first set where F reaches a fixed point in the sequence $\varnothing, F(\varnothing), FF(\varnothing), \ldots, F^{\omega}(\varnothing), FF^{\omega}(\varnothing), \ldots$, taking the union of all previous stages at limit ordinals.

Here is the approximation sequence for the associated set function of $q \vee \langle a \rangle p$:

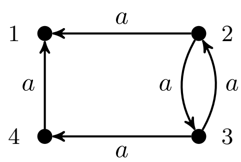
stage	*set*	*defining formula*
0	$\varnothing$	$\bot$
1	$\{5, 6\}$	q
2	$\{5, 6, 2\}$	$q \vee \langle a \rangle p$
3	$\{5, 6, 2\}$	the fixed point

What $\mu p \bullet (q \vee \langle a \rangle p)$ describes in general is the set of all points in the given model that can reach a q-world by a finite sequence of a-steps. In this way, the μ-calculus defines the typical modality $\langle a^* \rangle \varphi$ of propositional dynamic logic (PDL) used at many places in this book. ■

By a similar analysis, the formula $\mu p \bullet [a]p$ holds in points with only finite a-sequences coming out, the so-called "well-founded part" of the relation R_a. But there is an interesting difference. The approximation process for $\mu p \bullet (q \vee \langle a \rangle p)$ always stops in ω steps, while that for $\mu p \bullet [a]p$ can go on to any ordinal, depending on the size of the model. This has to do with the syntax of these fixed point formulas (cf. Fontaine 2010).

Related to the latter smallest fixed point formula is the greatest fixed point formula $\nu p \bullet \langle a \rangle p$ defining the s at which an infinite sequence $s R_a s_1 R_a s_2 R_a \cdots$ starts. Actually, $\nu p \bullet \varphi(p)$ is equivalent to $\neg \mu p \bullet \neg \varphi(\neg p)$.

Example 14.8 Evaluating greatest fixed point formulas
Consider the formula $\nu p \bullet \langle a \rangle p$ in the following model:



The computation stabilizes at the set of worlds $\{2, 3\}$. ■

Our syntax even allows inhomogeneous nested fixed point formulas of shapes such as $\nu p \bullet \mu q \bullet \varphi(p, q)$, whose intuitive meaning can be harder to decode.

Infinite evaluation games Niwinski & Walukiewicz (1996) and Stirling (1999) define evaluation games for the μ-calculus (Venema 2007 has a lucid presentation). These involve a significant change reflecting the ordinal approximation process, taking us into the realm of Chapter 5: runs may become infinite. Accordingly, the winning convention changes in a delicate manner that goes back to fundamental results connecting logic and automata theory (Rabin 1968, Thomas 1992).

DEFINITION 14.4 Evaluation games for the μ-calculus
In *μ-calculus evaluation games*, verifier and falsifier play by the earlier rules when the main operator is modal. When a fixed point formula $\mu p \bullet \varphi(p)$ or $\nu p \bullet \varphi(p)$ is reached in the game, the next formula is $\varphi(p)$, with the following understanding. Whenever later play hits an atom p, no test takes place (p is a bound variable), but $\mu p \bullet \varphi(p)$ (or $\nu p \bullet \varphi(p)$, as the case may be) is substituted back in.[177]

Next, we note that in infinite runs, some fixed point subformula of the finite initial formula φ must have been called infinitely often. Indeed, it is easy to see that there is a unique recurrent fixed point subformula occurring in the highest syntactic position in φ. We say that an infinite run is a *win for* $\boldsymbol{V}$ in the φ-game if the syntactically highest recurrent fixed point subformula is a greatest fixed point. If it is a smallest fixed point, then $\boldsymbol{F}$ wins. ■

The winning convention displays a general pattern from graph games with a "parity condition" (see Section 18.6 in Chapter 18, and Venema 2007).

As in our earlier analyses, a success lemma connects our language and these infinite games.

FACT 14.6 A formula in the modal μ-calculus is true if and only if verifier has a winning strategy in the game just described.

We refer to the cited literature for a proof, and its further background in automata theory. The μ-calculus is a good framework for studying general interactive processes, and its game aspects will return in Chapter 18.

We now move to a more general formalism showing the same ideas at work, which has also been used earlier on in this book, for instance, in Chapters 2, 8, and 13.

First-order fixed point logic First-order logic with fixed point operators LFP(FO)(Moschovakis 1974, Ebbinghaus & Flum 1999) was used for analyzing game-theoretic solution methods in Part II. Again the language has smallest fixed point operators

$$\mu P \bullet \varphi(P) \qquad \text{with } P \text{ occurring only positively in } \varphi$$

177 Here we assume, without loss of generality, that occurrences of fixed point variables have been made unique.

More precisely, with $\boldsymbol{x}$ (or $\boldsymbol{d}$) standing for finite tuples of variables (or objects),

$[\mu P, \boldsymbol{x} \bullet \varphi(P)](\boldsymbol{d})$ says that $\boldsymbol{d}$ is in the smallest set X with $\varphi^{\boldsymbol{M}}(X) = X$
$[\nu P, \boldsymbol{x} \bullet \varphi(P)](\boldsymbol{d})$ says that $\boldsymbol{d}$ is in the largest set X with $\varphi^{\boldsymbol{M}}(X) = X$

EXAMPLE 14.9 Transitive closure
For instance, $\mu P, xy \bullet (Rxy \vee \exists z(Rxz \wedge Rzy))$ is a definition for the transitive closure of a relation R, showing the typical recursive behavior of this notion:

$$trans(R)(x, y) \leftrightarrow R(x, y) \vee \exists z(R(x, z) \wedge trans(R)(z, y))$$

As with the μ-calculus, this recursive unwinding shows how games for LFP(FO) differ from first-order ones: they can *cycle*. For, if $\boldsymbol{V}$ chooses the right-hand disjunct, taking an R-successor, the original formula returns, and play loops. For certain types of formula, infinite games then become indispensable. ■

The games that we need are as before, with some notational adaptations.

DEFINITION 14.5 First-order fixed point games
In games for LFP(FO), verifier and falsifier play by earlier rules when the main operator is first-order. When a fixed point formula $[\mu P, \boldsymbol{x} \bullet \varphi(P)](\boldsymbol{d})$ is reached, the next formula is $\varphi(P)](\boldsymbol{d})$, with the following understanding. Whenever subsequent play reaches an atom $P\boldsymbol{e}$, no test takes place in the model (P is a bound variable), but $[\mu P, \boldsymbol{x} \bullet \varphi(P)](\boldsymbol{e})$ is substituted back in. Greatest fixed point formulas $[\nu P, \boldsymbol{x} \bullet \varphi(P)](\boldsymbol{d})$ are treated analogously.

Again, in infinite runs, some fixed point subformula of the finite initial formula φ must have been called infinitely often, and there is a unique subformula of this kind occurring in the highest syntactic position in φ. As was stipulated for the μ-calculus, an infinite run is a *win for* $\boldsymbol{V}$ in the φ-game if the syntactically highest recurrent fixed point subformula is a greatest fixed point. If it is a smallest fixed point, then $\boldsymbol{F}$ wins.[178] ■

Again, the following success lemma shows how games and models connect.

178 Consider the smallest fixed point for transitive closure. To show that Pde, verifier may first choose the second disjunct, take some object f for z, and claim that Pfe. But verifier may only do that finitely often; otherwise, a loss results. Greatest fixed points are dual: falsifier must put up in some finite number of cycles, while it is fine for verifier to keep the cycle going.

FACT 14.7 A formula in first-order fixed point logic is true if and only if verifier has a winning strategy in the game just described.

See Ebbinghaus & Flum (1999) and Doets (1999) for proofs. There is a general result in the background here. While these games do not have an open winning condition for players in the sense of the Gale-Stewart Theorem of Chapter 5, their winning condition is still Borel, making the games determined by Martin's Theorem.

EXCURSION Second-order reformulation
An alternative finite version of this game translates formulas $[\mu P, \boldsymbol{x}] \bullet \varphi(\boldsymbol{d})$ inside out by equivalent second-order statements. The Tarski-Knaster definition of a smallest fixed point says that $\boldsymbol{d}$ satisfies every predicate that is a smallest pre-fixed point of the monotonic set operation for the positive formula φ, or in second-order terms, $\forall Q \left(\forall x(\phi(Q, \boldsymbol{x}) \to Q\boldsymbol{x}) \to Q\boldsymbol{d}\right)$. A standard evaluation game for these second-order formulas is of finite depth, although at the cost of letting players choose sets of (tuples of) objects. As before, there is an issue in which sense this is the same as our earlier fixed point game, and this time, a non-trivial proof is needed to show that players' powers do not change.

There is a deep literature on evaluation games for fragments of second-order logic, of which Rabin (1968) is a classic. Walukiewicz (2002) is an influential format for automata-based games for monadic second-order logic (first-order logic with added quantifiers over sets), which plays an important role in games and computation, witness our discussion of forcing logics in Chapter 5. These and other automata-based games are explained (with many relevant results) in an accessible manner in Zanasi (2012). A short overview will be found in Chapter 18.

Games with changing models Even for first-order logic, other variations are possible on standard evaluation games, making players perform different tasks. In contrast to object-picking moves, one may consider removing objects from a domain without replacement, or moves that change a domain or interpretation function, changing the playground by adding or subtracting objects and facts. Such variations mix the process of logical evaluation with model construction that will be considered in Chapter 16.

14.6 Conclusion

This concludes our tour of two-player evaluation games as a dynamic view on logical semantics. Most of these games had finite depth, but some are naturally infinite when the language contains fixed points, a natural correlate to the notion of game-theoretic equilibrium in this book. In addition, we found a number of general game-theoretic themes illustrating general topics in this book: namely, determinacy, game equivalence, game algebra of choice, switch, or composition, and the systematic importance of calculus of strategies. Many of these themes will return in the integrative Parts V and VI of this book.

14.7 Literature

The presentation in this chapter is largely from van Benthem (1999).

Hintikka (1973) is a source of the ideas, and an introduction supported by software is found in Barwise & Etchemendy (1999). A recent non-trivial extension to many-valued logics is Fermüller & Majer (2013). Mann et al. (2011) has a more formal development, continuing on to the more complex games with imperfect information found in Chapter 21. We have also pointed at the flourishing literature on related games for fixed point logics in computer science, that link up with automata theory, with references on the μ-calculus such as Venema (2006), and Janin & Walukiewicz (1995). Chapter 18 will provide more details.