1
Games as Processes: Definability and Invariance

We start in a very simple fashion. A game tree with nodes and moves is a model for a process with states and transitions, of a kind that has been studied in logic, computer science, and philosophy under names such as Kripke models or labeled transition systems. Such models invite the use of logical languages that define basic process structure in a perspicuous way, while representing intuitive patterns of reasoning about social scenarios. A further current use of logic in games viewed in this manner is the analysis of computational complexity for model checking desirable properties, but such computational concerns will be less prominent in this book. In this chapter, we will see that, in particular, modal logics do many useful jobs, making them a first candidate for a *language of games*. However, a further fundamental perspective from logic makes sense. A choice of language is at the same time a choice for a level of structural invariance between semantic structures. Accordingly, our second main theme is *structural invariance between games*, looking for natural levels of viewing interaction. In brief: "When are two games the same?" There is no consensus on this important issue, but we put forward some proposals. Finally, we state a brief conclusion, mention some key literature that went into the making of this chapter, and list some further directions that continue themes raised in the main text.

NOTE The games in this chapter and most of this book are normally taken to be *finite*. When it is important to include infinite games, we will say so explicitly (cf. Chapters 5, 14, 20, and others). Many notions and results in this book also apply to infinite games, but we will not pursue this generalization systematically.

CAVEAT We repeat an earlier warning. This chapter is not an introduction to modal logic, and the same is true for other logics in this book. The textbook van Benthem (2010b) covers most of the basics that will be needed.

1.1 Games as process graphs in modal logic

Extensive games as process graphs We start by defining our basic models, using a slight variation on the presentation in Osborne & Rubinstein (1994).

DEFINITION 1.1 Extensive games
An *extensive game* $\boldsymbol{G}$ is a tuple $(I, A, H, t, \{\leq_i\})$ consisting of (a) a set I of players, (b) a set of actions or moves A, (c) a set H of finite or infinite sequences of successive actions from A (the histories), closed under taking finite prefixes and infinite limits of histories, (d) a turn function t mapping each non-terminal history having a proper continuation in H to a unique player whose turn it is, and finally, (e) for each player i, a connected preference relation $h \leq_i h'$ (player i weakly prefers h' to h) on terminal histories. Dropping preference relations from an extensive game yields an *extensive game form* being just the underlying pure action structure. ■

Several things can be generalized here, say, allowing partial preference orders with incomparable outcomes, allowing simultaneous moves, or dropping limit closure to model a wider class of temporal processes. However, the basic format is the most widely used, and understanding it will facilitate logical generalizations later.

To readers from other fields, extensive game forms will exhibit a familiar structure: they are just multi-agent "process graphs" in logic and computer science with a universe of states and transitions. In this chapter, we start at the level of game forms, looking at action structure only. Preference will return in Chapter 2.

DEFINITION 1.2 Extensive game models
An *extensive game model* is a tree $\boldsymbol{M} = (S, M, \boldsymbol{turn}, \boldsymbol{end}, V)$ with a set of nodes S and a family M of binary transition relations for the available moves, pointing from parent to daughter nodes. There are two special properties of nodes: non-final nodes have unary proposition letters $\boldsymbol{turn}_i$ indicating the player whose turn it is, while $\boldsymbol{end}$ marks end nodes. On top of this, the valuation V serves to interpret other relevant properties of nodes, such as utility values for players, or more ad hoc features of game states. ■

Extensive game models, also called labeled transition systems or Kripke models, are natural models for a logical language. In this chapter, we will concentrate on the use of modal logics for extensive games, although we will also look at extensive games as models for temporal logics in Chapters 5 and 6.

Basic modal logic Extensive game trees support a standard modal language.

DEFINITION 1.3 Modal game language and semantics
Modal formulas are defined using the following inductive syntax rules for Boolean operations and modalities, where the p come from a set *Prop* of atomic proposition letters, and the a from a set Act of atomic action symbols:

$$p \mid \neg\varphi \mid \varphi \vee \psi \mid \langle a \rangle \varphi$$

Boolean conjunction $\wedge$ and implication $\rightarrow$, as well as the universal modality $[a]$ are defined in terms of these as usual. Modal formulas are interpreted at nodes s as local properties of game stages in the earlier models $\boldsymbol{M}$, in the format

$$\boldsymbol{M}, s \models \varphi \qquad \text{formula } \varphi \text{ is true at node } s \text{ in model } \boldsymbol{M}$$

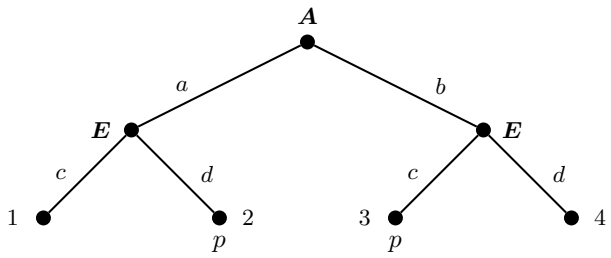
The inductive clauses of this truth definition are as usual for the Booleans, while there is the following key truth condition for labeled modalities:

$$\boldsymbol{M}, s \models \langle a \rangle \varphi \quad \text{iff} \quad \text{there exists some } t \text{ with } sR_at \text{ and } \boldsymbol{M}, t \models \varphi$$

This says that some particular instance of the move a is available at the current node leading to a next node in the game tree satisfying φ. The universal modality $[a]\varphi$ says that φ is true at all successor nodes reachable by an a-transition. ■

Modal operator combinations now describe possible interactions in games. We briefly repeat an example discussed at greater length in the Introduction.

EXAMPLE 1.1 Modal operators and strategic powers
Consider a simple two-step game like the following, between two players $\boldsymbol{A}$ and $\boldsymbol{E}$:



Player $\boldsymbol{E}$ clearly has a strategy for making sure that a state is reached where p holds, that is, a rule for responding to the opponent's moves producing some desired effect.

This feature of the game is directly expressed by a modal formula like this:

$$[a]\langle d\rangle p \wedge [b]\langle c\rangle p$$

Other modal formulas express other interesting properties of nodes in this game. More generally, we will see modal notations at work throughout this book. ■

As noted in the Introduction, notions like strategy and other items in game forms match their definition from game theory, unless explicitly noted otherwise.

Dynamic programs We can also introduce explicit notation for basic and complex moves in a game. For instance, letting the symbol *move* stand for the union of all relations available to players, in the preceding game, the modal combination

$$[\mathit{move}\text{-}\boldsymbol{A}]\langle \mathit{move}\text{-}\boldsymbol{E}\rangle\varphi$$

says that, at the current node, player $\boldsymbol{E}$ has a strategy responding to $\boldsymbol{A}$'s initial move which ensures that the property defined by φ results after two steps of play. Union is one of the program operations of an extension of the basic modal language called *propositional dynamic logic*, PDL (Harel et al. 2000).

Definition 1.4 PDL programs
Starting from atomic action symbols in a set *Act* as before, the *PDL programs* are defined by the following inductive syntax

$$a \mid \pi_1 \cup \pi_2 \mid \pi_1;\pi_2 \mid \pi^* \mid ?\varphi$$

of atomic actions, union, composition, finite iteration, and tests. In the above models, programs π are interpreted as binary transition relations R_π between nodes, according to the following inductive clauses:

$R_{\pi_1\cup\pi_2}$	$= R_{\pi_1} \cup R_{\pi_2}$	Choice as union of transition relations
$R_{\pi_1;\pi_2}$	$= R_{\pi_1} \circ R_{\pi_2}$	Sequential composition of relations
R_{π^*}	$= (R_\pi)^*$	Reflexive-transitive closure of relations
$R_{?\varphi}$	$= \{(s,t) \mid s = t \,\&\, \boldsymbol{M}, s \models \varphi\}$	Successful test of a property ■

All of these program operations make sense in the action structure of games, e.g., in defining strategies, and we will encounter them in many chapters to follow.

Excluded middle and determinacy Extending our observations to extensive games up to depth k, and using alternations $\Box\Diamond\Box\Diamond\cdots$ of modal operators up to length k, we can express the existence of winning strategies in any given finite game.

In this manner, standard logical laws acquire immediate game-theoretic import. In particular, consider the valid law of excluded middle

$$\Box\Diamond\Box\Diamond\cdots\varphi \vee \neg\Box\Diamond\Box\Diamond\cdots\varphi$$

or after some logical equivalences $\neg\Box = \Diamond\neg$, $\neg\Diamond = \Box\neg$ pushing negations inside:

$$\Box\Diamond\Box\Diamond\cdots\varphi \vee \Diamond\Box\Diamond\Box\cdots\neg\varphi$$

where the dots indicate the depth of the tree.

FACT 1.1 Modal excluded middle expresses the determinacy of finite games.

Here, determinacy is the basic property of win-lose games that one of the two players has a winning strategy. Since winning can be any condition φ of histories, in the above formula, the left-hand disjunct says that the responding player $\boldsymbol{E}$ has a winning strategy, and the right-hand disjunct that the starting player $\boldsymbol{A}$ has a winning strategy. (This may fail in infinite games: players cannot both have a winning strategy, but perhaps neither one has. Determined infinite games have been studied extensively in Descriptive Set Theory; Moschovakis 1980, Kechris 1994.)

Zermelo's theorem This brings us to an early game-theoretic result predating Backward Induction, proved by Ernst Zermelo in 1913 for zero-sum games, where what one player wins is lost by the other (win vs. lose is the typical example).

THEOREM 1.1 Every finite zero-sum two-player game is determined.

Proof Recall the algorithm in the Introduction for determining the player having the winning strategy at any given node of a game tree of this finite sort. The method worked bottom-up through the game tree. First, color end nodes *black* that are wins for player $\boldsymbol{A}$, and color the other end nodes *white*, being wins for $\boldsymbol{E}$. Then, if all children of node s have been colored already, do one of the following:

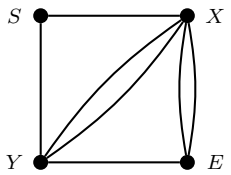
(a) If player $\boldsymbol{A}$ is to move, and at least one child is black: color s *black*; if all children are white, color s *white*.
(b) If player $\boldsymbol{E}$ is to move, and at least one child is white: color s *white*; if all children are black, color s *black*.

This algorithm eventually colors all nodes black where player $\boldsymbol{A}$ has a winning strategy, while coloring those where $\boldsymbol{E}$ has a winning strategy white. Once the root has been colored, we see the determinacy, and even who has the winning strategy.

The reason for the correctness of this procedure is that a player i whose turn it is has a winning strategy iff i can make a move to at least one daughter node where i has a winning strategy. ■

Application: The teaching game Zermelo's Theorem is widely applicable. A variant of the sabotage game in our Introduction shows another flavor of multi-move agent interaction.

EXAMPLE 1.2 Teaching, the grim realities
A student located at S in the following diagram wants to reach the escape E, whereas the teacher wants to prevent the student from getting there. Each line segment is a path that can be traveled. In each round of the game, the teacher first cuts one line, anywhere, and the student must then travel one link still open at the current position:



In this particular game, the teacher has a winning strategy, by first cutting a line to the right between X and E, and then waiting for the student's moves, cutting lines appropriately. General games like this arise on any graph with single or multiple lines. Gierasimczuk et al. (2009) provide links with real teaching scenarios. ■

Zermelo's Theorem explains why the student or the teacher has a winning strategy: cutting lines must end. Many games fall under this result, like most logic games in Part IV. For applications to computer games, see van den Herik et al. (2011).

Modal μ-calculus for equilibrium notions A good test for expressive power of logics is their capacity for representing basic arguments in the field under study. Our basic modal language cannot yet express the proof of Zermelo's Theorem. Starting from atomic predicates $\boldsymbol{win}_i$ at end nodes marking which player has won, that proof inductively defined new predicates WIN_i (i.e., player i has a winning strategy at the current node, where we use i and j to indicate the opposing players) through a recursion of the form

$$WIN_i \;\leftrightarrow\; (\boldsymbol{end} \wedge \boldsymbol{win}_i) \vee (\boldsymbol{turn}_i \wedge \langle \textit{move-}i \rangle\, WIN_i) \vee (\boldsymbol{turn}_j \wedge [\textit{move-}j]\, WIN_i)$$

Here *move-x* is the union of all moves for player x. Note how the defined expression WIN_i occurs recursively in the body of its own definition.

Despite its modal shape, this inductive definition has no explicit version in our modal base language. We need a logic that supports inductive definitions and complex recursions in computation. The *modal μ-calculus* extends modal logic with operators for "smallest fixed points"

$$\mu p \bullet \varphi(p)$$

where formulas $\varphi(p)$ must have a special syntax. The propositional variable p may occur only positively in $\varphi(p)$, that is, in the scope of an even number of negations.[1] This ensures that the following approximation function on sets of states:

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

is monotonic in the inclusion order:

$$\text{whenever } X \subseteq Y, \text{ then } F^M_\varphi(X) \subseteq F^M_\varphi(Y)$$

On complete lattices (such as power sets of models), the *Tarski-Knaster Theorem* says that monotonic maps F always have a "smallest fixed point," a smallest set of states X where $F(X) = X$. One can reach this smallest fixed point F_* through a sequence of approximations indexed by ordinals until there is no more increase:

$$\varnothing, F(\varnothing), F^2(\varnothing), \ldots, F^\alpha(\varnothing), \ldots F_*$$

The formula $\mu p \bullet \varphi(p)$ holds in a model $\boldsymbol{M}$ at just those states that belong to the smallest fixed point for the map $F^M_\varphi(X)$. Completely dually, there are also "greatest fixed points" for monotonic maps, and these are denoted by formulas

$$\nu p \bullet \varphi(p), \qquad \text{with } p \text{ occurring only positively in } \varphi(p).$$

Greatest fixed points are definable from smallest ones (and vice versa), as shown in the valid formula $\nu p \bullet \varphi(p) \leftrightarrow \neg \mu p \bullet \neg \varphi(\neg p)$, where $\neg \varphi(\neg p)$ still has p positive.

---

1 Alternately, $\varphi(p)$ has to be a formula constructed from arbitrary p-free formulas of the language plus occurrences of p, using only $\vee$, $\wedge$, $\Box$, $\Diamond$, and μ-operators.

These are just some of the basic properties of this system, which embodies the decidable modal core theory of induction and recursion. There is a fast-growing literature on the μ-calculus (cf. Bradfield & Stirling 2006, Venema 2007). This system will return in several parts of this book, including Chapters 14 and 18.

FACT 1.2 The Zermelo solution is definable as follows in the modal μ-calculus:

$$WIN_i = \mu p \bullet (\boldsymbol{end} \wedge \boldsymbol{win}_i) \vee (\boldsymbol{turn}_i \wedge \langle \textit{move-}i \rangle p) \vee (\boldsymbol{turn}_j \wedge [\textit{move-}j] p)$$

Proof The key points are the modal format of the inductive response clauses, plus the positive occurrences of the atom p standing for the predicate defined. ■

Fixed point definitions fit well with the equilibrium notions of game theory that often have some iterative intuition attached of successive approximation. Hence, the μ-calculus has many uses in games, as discussed further in Chapters 7 and 13 where we will use extended fixed point logics to analyze other equilibrium notions.[2]

Control over outcomes and forcing Winning is just one aspect of control in a game. Games are all about powers over outcomes that players can exercise via their strategies. This suggests further logical notions with a modal flavor.

DEFINITION 1.5 Forcing modalities $\{i\}\varphi$
$\boldsymbol{M}, s \models \{i\}\varphi$ iff player i has a strategy for the subgame starting at s that guarantees only end nodes will be reached where φ holds, whatever the other player does. ■

FACT 1.3 The modal μ-calculus can define forcing modalities for games.

Proof The modal fixed point formula

$$\{i\}\varphi = \mu p \bullet (\boldsymbol{end} \wedge \varphi) \vee (\boldsymbol{turn}_i \wedge \langle \textit{move-}i \rangle p) \vee (\boldsymbol{turn}_j \wedge [\textit{move-}j] p)$$

defines the existence of a strategy for player i making proposition φ hold at the end of the game, whatever the other player does.[3] ■

---

2 In infinite games, it seems better to use greatest fixed points defining a largest predicate satisfying the recursion. This does not build strategies inductively from below, but views them as rules that can be called, and always remain at our service. This is the perspective of co-induction in *co-algebra* (Venema 2006). Chapters 4, 5, and 18 discuss such strategies.

3 One can easily modify the definition of $\{i\}\varphi$ to enforce truth of φ at all intermediate nodes, a variant that will be used in Chapter 5.

Analogously, shifting modalities slightly, the formula

$$COOP\varphi = \mu p \bullet (\boldsymbol{end} \wedge \varphi) \vee (\boldsymbol{turn}_i \wedge \langle \textit{move-i} \rangle p) \vee (\boldsymbol{turn}_j \wedge \langle \textit{move-j} \rangle p)$$

defines the existence of a cooperative outcome φ. However, this notion is already definable in PDL as well, using the program modality

$$\langle \big((?\boldsymbol{turn}_i; \textit{move-i}) \cup (?\boldsymbol{turn}_j; \textit{move-j}) \big)^* \rangle (\boldsymbol{end} \wedge \varphi)$$

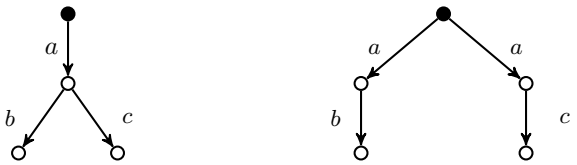
The program is an explicit strategy that makes the forcing statement true. We will say more on the issue of explicit strategies versus forcing modalities in Chapter 4.

1.2 Process and game equivalences

We have seen how many properties of game forms can be defined in modal formulas, making modal logic a good medium for reasoning about the basics of interaction. But next to language design, there is another basic perspective on games. To see this, recall the view of games as processes as explored in our Introduction.

Process equivalence Process graphs represent processes, and a natural question arises of when two graphs stand for the same process. As elsewhere in mathematics, we want an appealing invariance relation. When are two processes the same?

EXAMPLE 1.3 The same process, or not?
We repeat two pictures of processes, or machines, from our Introduction:



Both diagrams produce the same finite traces of observable actions $\{ab, ac\}$, so qua input-output behavior the machines are the same under "finite trace equivalence." But in doing so, the first machine starts deterministically, and the other with an internal choice. In order to explore such internal workings of a process, the measure must have a finer structural comparison distinguishing these models. ■

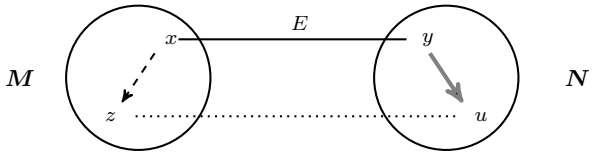
Bisimulation As will be clear from the example, there is no unique answer: different structural invariances may set natural levels of process structure. Finite trace equivalence is one extreme, another extreme is the standard notion of isomorphism, preserving every detail of size and relational structure of a process that is definable in first- or higher-order languages. For present purposes, however, we want to be in between these two levels. The structural invariance matching the expressive power of modal logic is a notion that has been proposed independently in many settings, including computer science and set theory. It works on any graph model of actions and states, providing an account of "simulating" (the key notion in relating different processes) both observable actions and internal choices that led to these.

Definition 1.6 Bisimulation
A *bisimulation* is a binary relation E between states of two graphs $\boldsymbol{M}$ and $\boldsymbol{N}$ such that, if xEy, then we have (1) atomic harmony, and (2) back-and-forth clauses,

(1) x, y verify the same proposition letters.

(2a) If xRz, then there exists u in $\boldsymbol{N}$ with yRu and zEu.

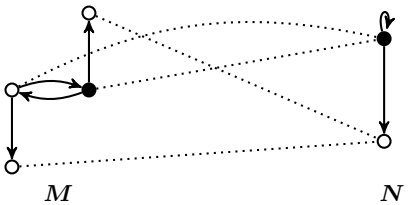
(2b) If yRu, then there exists z in $\boldsymbol{M}$ with xRz and zEu.



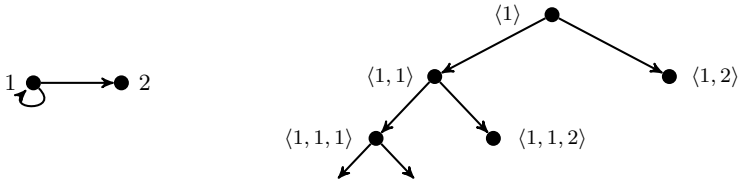
■

Bisimulation respects local properties plus options available to the process at any stage. Our earlier two finite-trace equivalent graphs are not bisimilar in their roots. Before we continue with exploring its properties, let us point out some major uses.

Example 1.4 Uses of bisimulation
Bisimulation contracts process graphs to a simplest equivalent graph, as in:



A reverse use of bisimulation unravels any process graph $\boldsymbol{M}, s$ to a rooted tree. The states are all finite paths through $\boldsymbol{M}$ starting with s and passing on to R-successors at each step. One path has another path accessible if the second is one step longer. The valuation on paths is copied from the value on their last nodes.



An unraveling is like a game tree, with the branches encoding all possible runs. This format is very convenient for theoretical purposes in logic and computation.

Invariance of modal formulas Bisimulation preserves the truth of modal and dynamic formulas across models, and there are converses, too. The following are the basic model-theoretic facts in this area.

THEOREM 1.2 For all graphs $\boldsymbol{M}$ and $\boldsymbol{N}$ with nodes s, t, condition (a) implies (b):

(a) There is a bisimulation E between $\boldsymbol{M}$ and $\boldsymbol{N}$ with sEt.

(b) $\boldsymbol{M}, s$ and $\boldsymbol{N}, t$ satisfy the same formulas of the μ-calculus.

This implies invariance of modal and dynamic formulas under bisimulation. The following partial converse says that the basic modal language and the similarity relation match on finite models.

THEOREM 1.3 For any two finite models $\boldsymbol{M}$ and $\boldsymbol{N}$ with given nodes s and t, the following statements are equivalent:

(a) $\boldsymbol{M}, s$ and $\boldsymbol{N}, t$ satisfy the same modal formulas.

(b) There is a bisimulation E between $\boldsymbol{M}$ and $\boldsymbol{N}$ with sEt.

Our third result says that, at this same level of description, the dynamic language even provides complete descriptions for any finite graph.

THEOREM 1.4 For each finite graph $\boldsymbol{M}$ and node s, there exists a dynamic logic formula $\delta(\boldsymbol{M}, s)$ such that the following are equivalent for all graphs $\boldsymbol{N}, t$:

(a) $\boldsymbol{N}, t \models \delta(\boldsymbol{M}, s)$.

(b) $\boldsymbol{N}, t$ is bisimilar to $\boldsymbol{M}, s$.

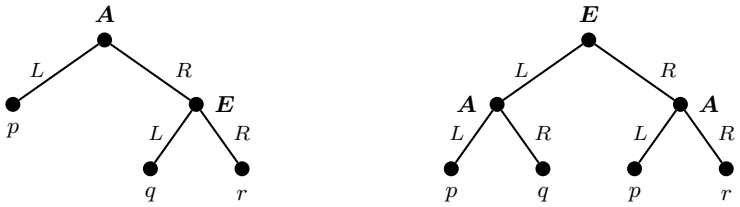
Proofs of all this can be found in van Benthem (2010b). The last two facts even hold for arbitrary graphs when we increase the expressive power of basic modal logic with arbitrary infinite conjunctions and disjunctions in its construction rules.[4]

Hierarchy of languages and process equivalences There is a hierarchy of natural process equivalences, from coarser ones like finite trace equivalence to finer ones like bisimulation. No best level of sameness exists in process theory: it depends on the purpose. This latitude is also known from studies of space, from fine-grained in geometry to coarse-grained in topology: it all depends on what you mean by "is." Crucially, the same seems true for games. Extensive games match well with bisimulation. However, our earlier strategic power perspective was closer to an input-output process view such as trace equivalence.

Hierarchy of languages The ladder of structural simulations has a syntactic counterpart. The finer the process equivalence, the more expressive a matching language for the relevant process properties. In this setting, the previously discussed modal results generalize. Similar invariance and definability results hold for many kinds of process equivalence (van Benthem 1996). Further, we submit that the same is true for games.

Game equivalence and invariances The same style of invariance thinking applies to the question when two games are the same (van Benthem 2002a).

EXAMPLE 1.5 Local versus global game equivalence
Recall an earlier example from the Introduction. Consider the following two games:



Are these two games the same? The answer depends on our level of interest. In terms of turns and local moves, the games are not equivalent, and some kind of bisimulation seems the right invariance, whose absence is matched by a difference

4 Bisimulation also applies to infinite games with infinite histories as outcomes. In this case, matching languages are the branching temporal logics of Chapters 5, 6, and 9.

that is expressible in a modal language. For instance, the game on the right, but not that on the left, satisfies the modal formula $\langle L\rangle\langle R\rangle \mathit{true}$ at its root. ■

More generally, in line with our definability results for process graphs, game forms can be described either via structural equivalences or in terms of game languages with logical formulas (Bonanno 1993, van Benthem 2001b).

Power equivalence and forcing modalities Let us explore the variety. If our focus is on achievable outcomes of the two games only, then the verdict changed.

EXAMPLE 1.5, CONTINUED
Both players have the same powers of achieving outcomes in both games:

(a) $\boldsymbol{A}$ can force the outcome to fall in the sets $\{p\}$, $\{q, r\}$.
(b) $\boldsymbol{E}$ can force the outcome to fall in the sets $\{p, q\}$, $\{p, r\}$.

As in the Introduction, a player's powers are those sets U of outcomes for which the player has a strategy making sure the game will end inside U, no matter what others do. In the game on the left, player $\boldsymbol{A}$ has strategies *left* and *right*, yielding powers $\{p\}$ and $\{q, r\}$, and player $\boldsymbol{E}$ has two strategies yielding powers $\{p, q\}$ and $\{p, r\}$. On the right, $\boldsymbol{E}$ has *left* and *right*, giving the same powers as on the left. But $\boldsymbol{A}$ now has four strategies:

$$\textit{left}\colon L,\ \textit{right}\colon L, \quad \textit{left}\colon L,\ \textit{right}\colon R, \quad \textit{left}\colon R,\ \textit{right}\colon L, \quad \textit{left}\colon R,\ \textit{right}\colon R$$

The first and fourth strategies give the same powers for $\boldsymbol{A}$ as on the left, while the second and third strategies produce weaker powers subsumed by the former. ■

We will return to these simple but important examples at many places in this book. Again, there is a matching notion of power bisimulation plus a modal language for this level of description, involving the strategic forcing modalities $\{i\}\varphi$ of Section 2.1 of Chapter 2. This forcing perspective will be explored at greater length in Chapters 11 and 19. It is also the intuitive equivalence view behind the logic games that we will study in Part IV.

REMARK Levels and transformations in game theory
Issues of grain size have long been studied in game theory. In particular, extensive games induce a notion of extracted information that can be captured by *transformations* between games having the same normal or reduced form (cf. Thompson

1952, Kohlberg & Mertens 1986).[5] On a different tack, Bonanno (1992a) identifies extracted information with a set-theoretic form close to forcing, and gives a matching transformation of interchange of contiguous simultaneous moves.

1.3 Further notions of game equivalence

Having introduced our two main topics of definability in logical languages for games and matching notions of structural simulation, we now explore a few variations, some getting closer to the game-theoretic literature.

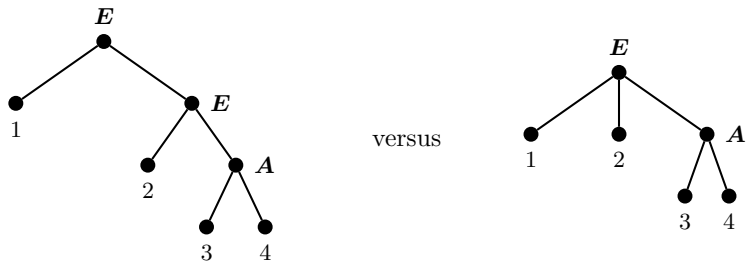
Alternating bisimulation Are there other natural game equivalences beyond the present two? Finite trace equivalence seems too coarse, but bisimulation sometimes seems too fine. The following two non-bisimilar single-player games seem equivalent:

One does not normally distinguish players' internal actions this finely, and the switch is in fact one of the "Thompson transformations" of Thompson (1952). But things matter with switches between different players, moving into another zone of control. We might not call game shapes equivalent if the turn patterns were very different, as with the earlier game, but now for formulas $(A \vee B) \wedge C$ and $A \vee (B \wedge C)$. Formulating a matching notion calls for a mixture of our earlier ideas.

Definition 1.7 Alternating bisimulation
An *alternating bisimulation* only requires the zigzag conditions of bisimulation with respect to "extended moves" that consist of a finite action sequence by one player ending in a state that is a move for the other player, or an endpoint. Also, it disregards the exact nature of these moves, viewing them solely as transitions. ■

5 Transformations are close to mathematical invariances in geometry and other fields. The closest analogue in our setting are bisimulations, although these are not functions. One associated transformation on models is the earlier bisimulation contraction.

EXAMPLE 1.6 Compacting games
Alternating bisimulation will identify the following two game trees:



This allows for normalizing games so that each move leads to a turn change. ■

Alternating bisimulation can again be tied up with matching modal game languages, but no complete characterization seems to be known. It resembles the intermediate forcing bisimulations of Chapter 11 that record players' powers for getting to any position of a game.[6]

Simulating strategies A different intuition about game equivalence that many people share might be phrased as follows:

players should have matching strategies in both games.

Ordinary bisimulation implies this property of matching.

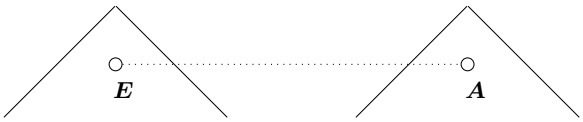
FACT 1.4 Games that admit a bisimulation between their roots allow all players to copy strategies for forcing sets of end nodes.

The reason is that the zigzag conditions allow both players to copy moves across. But bisimulation seems too strong, and weaker invariances suffice, depending on how the matching is spelled out. So far, there seems to be no best formulation of game equivalence as having the same strategies modulo effective simulation.

DIGRESSION Perhaps we must think differently here, using two simulation relations, one for player $\boldsymbol{E}$ and one for $\boldsymbol{A}$, comparing their separate powers across nodes in two games. When turns are the same, ordinary zigzag clauses for available moves seem acceptable. But the situation gets harder when the turns are not the same:

---

6 It is also related to stuttering bisimulations in computer science (Gheerbrant 2010).



With such a turn mismatch, single moves by player $\boldsymbol{E}$ must correspond to responses to all of $\boldsymbol{A}$'s moves on the other side. But $\boldsymbol{E}$'s powers on the left need not match her powers at each $\boldsymbol{A}$-successor on the right: a counterexample is easily found in our propositional distribution games. We may need inclusion of players' powers via "directed simulations" (Kurtonina & de Rijke 1997).

Links with logic games Game comparison also makes sense for the special realm of logic games in Part IV. But logic games also add to the current style of analysis. In Chapter 15, we will study Ehrenfeucht-Fraïssé games that test two given models for similarity over some specified number of rounds. These games add fine structure to notions of simulation, measuring differences that can be detected by concrete modal or first-order formulas. In this finer perspective, a bisimulation is a global winning strategy for establishing similarity in a game with an infinite number of rounds. Thus, logic games will provide a way of refining our earlier question of structural equivalence to the question of:

How similar are two given games?

Different levels once more Different game equivalences need not stand for deep competing modeling choices; instead, they may just reflect a frequent phenomenon in practice. The same game $\boldsymbol{G}$ can often be described at different levels, as one can individuate moves more or less finely. In the limit, we can even form a kind of extreme coarsening $\mathit{coarse}(\boldsymbol{G})$ by keeping the old nodes, but taking just two relations for the players as follows:

R_E is the union of all moves for player $\boldsymbol{E}$, and R_A is defined likewise.

Since $\mathit{coarse}(\boldsymbol{G})$ has less information than $\boldsymbol{G}$ itself, it does not support a reduction of properties of $\boldsymbol{G}$: things go the other way around. To see this, take any modal formula φ in the language of $\mathit{coarse}(\boldsymbol{G})$, and translate it into a formula $\mathit{fine}(\varphi)$ replacing the new relation symbols by their definitions. This results in the equivalence

$$\mathit{coarse}(\boldsymbol{G}), s \models \varphi \quad \text{iff} \quad \boldsymbol{G}, s \models \mathit{fine}(\varphi)$$

A related view in Chapter 19 will distinguish between games themselves and "game boards" that games are played on. One level is an external setting for observable effects of the game. The other is a richer game structure with internal predicates, such as turns, preferences, wins, and losses. A typical connection between the two ways of viewing one and the same game is the "adequacy lemma" for evaluation games in the Introduction that stated an equivalence between

$$\text{(a) } \boldsymbol{M}, s \models \varphi, \quad \text{(b) } \boldsymbol{game}(\boldsymbol{M}, \varphi), \langle s, \varphi \rangle \models \mathit{WIN}_{\boldsymbol{V}}$$

This relates a statement about powers of players in a game with properties of its game board. Two-level views are often illuminating. In particular, the above links may be used to analyze the complexity of strategies in games. We postpone a further analysis of multiple views on games until Parts III and V of this book.

1.4 Conclusion

The main points We have found our first serious connection between logic and games. Extensive games are processes of a kind well-known in computer science, and for a start, we have shown how modal languages fit well for the purpose of defining game-theoretic notions and capturing basic game-theoretic reasoning. We gave a systematic family of languages (modal logic, dynamic logic, μ-calculus) that link game theory with computational logic, allowing for traffic of ideas. In particular, modal languages correlate with natural notions of structural invariance, reflected in different notions of process equivalence. Our second main point was taking the same invariance thinking to games, with levels ranging from finer to coarser, a common mathematical perspective for defining a family of structures.

Benefits By paying systematic attention to links between structure and language, logic brings to light key patterns in definitions and inferences, such as the modal quantifier interchange underlying strategic behavior. Moreover, one can use this framework for model-checking given games for specific properties, proving completeness for calculi of reasoning about interaction, determining computational complexity of game-theoretic tasks, or investigating model-theoretic behavior such as transfer of properties from one game to another. In this book, we will not pursue such applications in any technical detail, but they are there, and we will continue to develop many further relevant interfaces.

Open problems One benefit of a logical stance is new problems. We conclude with some open problems raised by the analysis in this chapter. What are the best logical languages for formalizing basic game-theoretic proofs and defining major structures in games? We have suggested that a modal language, perhaps with fixed point operators, is a suitable vehicle, but is this borne out with further game-theoretic phenomena? This question of fit to basic reasoning about interaction will be addressed in the chapters to follow in Parts I and II. Related to this issue of language design is another: What are the most natural notions of structural equivalence between games? As we have seen, there are several natural levels here, and none of the invariances that we have mentioned seems to exhaust the topic.

1.5 Literature

This chapter is based on the process view of games from van Benthem (2002a).

There is a broad literature on logics for basic game structure. Pioneering papers on logical languages for games and their extracted information content are Bonanno (1992a, 1993), and a good survey of many varieties of game logics is the chapter van der Hoek & Pauly (2006) in the *Handbook of Modal Logic*. Structural equivalences, view levels, and transformations in game theory were discussed in famous papers such as Thompson (1952) and Kohlberg & Mertens (1986). Further interfaces between logic and game theory will occur throughout Parts I and II of this book, but also in Chapters 12 and 13 on logics for strategic games.

2
Preference, Game Solution, and Best Action

Real games arise over bare game forms when we take into account how players *evaluate* possible outcomes. Available actions give the kinematics of what can happen in a game, but it is only their interplay with evaluation that provides a more explanatory dynamics of well-considered intelligent behavior. How players evaluate outcomes is encoded in utility values, or in preference orderings.

Logic has long been applied to preference structure. In this chapter, we first review some basic preference logic from philosophy and computer science, showing how it applies to games. Then we analyze the paradigmatic solution procedure of Backward Induction as a pilot case, focusing on two features: (1) the role of *rationality* as a bridge law between information, action, and preference, and (2) the role of *recursive approximation* to the optimal outcome. There need not be a unique best level of syntactic detail for such a conceptual analysis, and we will present two levels of zoom. We first define the Backward Induction solution in a first-order fixed point logic of action and preference, making a junction with the area of computational logic. Next, we hide the recursive machinery, and use a modal logic to study the basic properties of "best actions" as supplied by the Backward Induction procedure. As usual, the chapter ends with a statement of further directions, conclusions, and open problems, as well as some selected literature behind the results presented here.

2.1 Basic preference logic

Models To model preferences, we start with a simple setting that lies behind many decision problems, games, and other scenarios (cf. Hanson 2001, Liu 2011).

DEFINITION 2.1 Preference models
Preference models $\boldsymbol{M} = (W, \leq, V)$ are standard modal structures with a set of worlds W standing for any sort of objects that are subject to evaluation and comparison, a binary betterness relation $s \leq t$ between worlds (i.e., t is at least as good as s), and a valuation V for proposition letters encoding unary properties of worlds or other relevant objects. ■

The comparison relation $\leq$ will usually be different for different agents, but in defining the basic logic, we will suppress agent subscripts $\leq_i$ for greater readability. We use the artificial term betterness to stress that this is an abstract comparison relation, making no claim yet concerning the intuitive term preference. Note that we are comparing individual worlds here, not properties of worlds. In actual parlance, preference often runs between generic properties of worlds or events, as in preferring tea over coffee. We will see in a moment how the latter view can be dealt with, too.

Very similar models with plausibility orderings are used for modeling belief (Girard 2008, van Benthem 2007c), and there are also connections with conditional logic and non-monotonic logics. These links will return in Part II of this book.

Constraints on betterness orders Which properties should a genuine betterness relation have? Total orders satisfying reflexivity $\forall x : x \leq x$, transitivity $\forall xyz : ((x \leq y \wedge y \leq z) \rightarrow x \leq z)$, and connectedness $\forall xy : (x \leq y \vee y \leq x)$, are common in decision theory and game theory, as these properties are induced by agents' numerical utilities for outcomes. But in the logical and philosophical literature on preference, a more general medium has been proposed, too, of *pre-orders*, satisfying just reflexivity and transitivity. Then there are four intuitively irreducible basic relations between worlds:

$w \leq v$, $\neg v \leq w$	$(w < v)$	w strictly precedes v
$v \leq w$, $\neg w \leq v$	$(v < w)$	v strictly precedes w
$w \leq v$, $v \leq w$	$(w \sim v)$	w, v are indifferent
$\neg w \leq v$, $\neg v \leq w$	$(w \# v)$	w, v are incomparable

The latter two clauses, although often confused, describe different situations.

One can also put two relations in betterness models: a "weak order" $w \leq v$ for at least as good, and a "strict order" $w < v$ for 'better', defined as $w \leq v$ and $\neg v \leq w$, respectively. This setup fits belief revision (Baltag & Smets 2008) and preference merge (Andréka et al. 2002). The logic of this extended language is axiomatized in van Benthem et al. (2009c), and related to the philosophical literature.

Modal logics Our base models interpret a standard modal language. In particular, a modal formula $\langle\leq\rangle\varphi$ will make the following local assertion at a world w:

$$\boldsymbol{M}, w \models \langle\leq\rangle\varphi \quad \text{iff} \quad \text{there exists a } v \geq w \text{ with } \boldsymbol{M}, v \models \varphi$$

that is, there is a world v at least as good as w that satisfies φ. In combination with other standard modal operators, in particular, a *universal modality* $U\varphi$ saying that φ holds at all worlds, this formalism can express quite a few further notions.

EXAMPLE 2.1 Defining conditionals from preference
Consider the bimodal formula

$$U\langle\leq\rangle[\leq]\varphi$$

This says that everywhere, there is some better world upward from which φ holds. In finite pre-orders, this says that all maximal elements in the ordering (having no properly better worlds) satisfy φ. But then we are close to other basic notions turning on maximality. Boutilier (1994) showed how a preference modality can define conditionals $\psi \Rightarrow \varphi$ in the style of Lewis (1973) with the following combination:

$$U(\psi \rightarrow \langle\leq\rangle(\psi \wedge [\leq](\psi \rightarrow \varphi)))$$

This is simply the standard ψ-relativized form of $U\langle\leq\rangle[\leq]\varphi$, saying, at least in finite models, that φ is true in all maximal worlds satisfying ψ. ■

Later on, we will use this modal language to define the Backward Induction solution for extensive games. By doing things this way, we can use the standard machinery of modal deduction to analyze the behavior of conditionals, or game-theoretic strategies.

The modal base logic of preference pre-orders is the system $S4$, while connectedness validates $S4.3$. In general, assumptions on orderings induce modal axioms by the standard technique of frame correspondences (cf. Blackburn et al. 2001).

Propositional preference The modal language describes local properties of betterness at worlds. But a betterness relation need not yet determine what we mean by agents' preferences in a more colloquial sense. Many authors consider preference to be a relation between propositions calling for comparison of sets of worlds. For a given relation $\leq$ among worlds, this may be achieved by "set lifting" of relations given on points. One ubiquitous proposal in such lifting is the $\forall\exists$ stipulation that

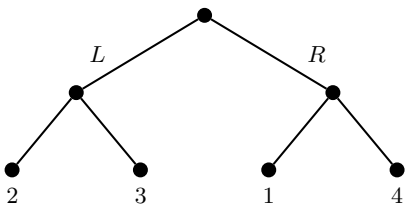
$$\forall\exists\text{-}\textit{rule} \quad \text{a set } Y \text{ is preferred to a set } X \text{ if } \forall x \in X\, \exists y \in Y : x \leq y$$

However, alternatives are possible. Von Wright's pioneering view of propositional preference is analyzed in van Benthem et al. (2009c) as the ∀∀ stipulation that

$\forall\forall$-*rule* a set Y is preferred to a set X if $\forall x \in X\, \forall y \in Y : x \leq y$

Liu (2011) reviews proposals for set lifted relations in various fields, but concludes that no consensus on one canonical notion of preference seems to have ever emerged. Preference as a comparison between propositions may depend on the scenario. For instance, in a game, when comparing sets of outcomes that can be reached by available moves, players have options. They might prefer a set whose minimum utility value exceeds the maximum of another: this is like the ∀∀ reading, which can produce incomparable nodes. But it would also be quite reasonable to require that the maximum of one set exceeds the maximum of the other, producing a connected order that would be rather like the ∀∃ reading.

Example 2.2 Options in set lifting
Consider the choice between move L and move R in the following:



The ∀∃ reading prefers R over L, the ∀∀ reading makes them incomparable, while maximizing minimal outcomes would lead to a preference for L over R. ■

Modal logics again Many lifts are definable in a modal base logic, with a few extra gadgets. For instance, using the universal modality $U\varphi$ (saying that φ is true in all worlds), with formulas standing for sets of worlds, ∀∃ preference is expressed by $U(\varphi \to \langle\leq\rangle\psi)$. For ∀∀ preference, things are more complex: see van Benthem et al. (2009c) for a proposed solution.[7]

7 An alternative view in Jiang (2012) defines set preferences by a process of comparing samples, and presents a dynamic logic of object picking.

2.2 Relational strategies and options for preference

Strategies as subrelations of the move relation To make our models of preference fit with games, we make a few changes from the usual setup. A strategy is usually taken to be a function on nodes in a game tree, yielding a unique recommendation for play. But in many settings, it makes sense to think of strategies as nondeterministic binary subrelations of the total relation *move* (the union of all actions in the game) that merely constrain further moves by marking one or more as admissible. This reflects a common sense view of strategies as plans for action, and it facilitates defining strategies in dynamic logic (cf. Chapter 4). Our numerical version of Backward Induction already had this relational flavor. Its computed relation *bi* linked nodes to all daughters with maximal values for the active player, of which there may be more than one.[8]

Solution algorithms and notions of preference Here is another important point about what look like obvious rules of computation: they embody *assumptions about players*. Recall that our Backward Induction clause for the non-active player took a minimal value. This is a worst-case assumption that the active player does not care at all about the other player's interests. But we might also assume some minimal cooperation, choosing maximal values for the other player among the maximal nodes. This variety of versions highlights an important feature: solution methods are not neutral, they tend to encode significant views of a game. Here is another way of phrasing this: things depend on what we mean by rationality.

Rationality: Avoiding stupid moves An active player must compare different moves, each of which, given the relational nature of the procedure, allows for many leaves that can be reached via further *bi*-play. A minimal notion of rational choice says that

> I do not play a move when I have another move whose outcomes I prefer.

8 Different views of strategies have been discussed in game theory in terms of plans of action, recommendations for action, or predictions of actions. Different choices may favor relational or functional views (cf. Greenberg 1990, Bonanno 2001). In this chapter, we will mainly follow the recommendation view, although Chapter 8 also casts strategies as beliefs or expectations about future behavior.

This seems plausible, but what notion of preference is meant here? In our first Backward Induction algorithm, a player i preferred a set Y to X if the minimum of its values for i is higher. This is the earlier $\forall\exists$ pattern for a set preference

$$\forall y \in Y\, \exists x \in X : x \leq_i y$$

However, the same notion of rationality allows for alternatives. A common notion of preference for Y over X that we saw already is the $\forall\forall$ view that

$$\forall y \in Y\, \forall x \in X : x \leq_i y$$

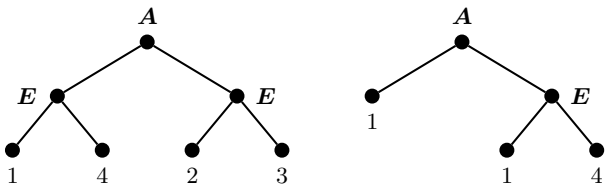
Relational Backward Induction The latter view suggests a minimal version of game solution where players merely avoid "strictly dominated" moves that are worse no matter what (Osborne & Rubinstein 1994). This will be our running example.

EXAMPLE 2.3 Relational Backward Induction
Call a move *a dominated* if it has a sibling move all of whose reachable endpoints are preferred by the current player to all reachable endpoints via a itself.

Now, first, mark all moves as active. The algorithm works in stages. At each stage, mark dominated moves in the $\forall\forall$ sense of set preference as passive, leaving all others active. In this comparison, reachable endpoints by an active move are all those that can be reached via moves that are still active at this stage. ■

This is a cautious notion of game solution making weaker assumptions about the behavior of other players than our earlier version.[9] We write bi for the subrelation of the total *move* relation produced at the end.

EXAMPLE 2.4 Some comparisons
Consider the following games, where the values indicated are utilities for player ***A***. For simplicity, we assume that player ***E*** has no preference between ***E***'s moves:



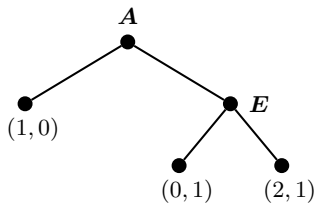
9 Many versions agree on so-called "distinguished games" that separate all histories.

In the game on the left, our original Backward Induction algorithm makes $\boldsymbol{A}$ go right, as the minimum 2 is greater than 1. But cautious *BI* accepts both moves for $\boldsymbol{A}$, as none strictly dominates the other. This may be viewed as set-lifted preference for very risk-averse players.

Interestingly, both versions pass all moves in the game to the right. This seems strange, as $\boldsymbol{A}$ might go right at the start, having nothing to lose, and a lot to gain. But analyzing all variants for preference comparisons between moves is not our goal here, and we move on to other topics. ■

EXAMPLE 2.5 More comparisons

Different views of Backward Induction also emerge when we think of Nash equilibria for *functional strategies*. Consider the following game:



Our cautious Backward Induction computation allows all moves in this game, since no move is dominated. However, there are two Nash equilibria in functional strategies: (L, L) and (R, R), corresponding to pessimistic or optimistic views on $\boldsymbol{A}$'s part as to what $\boldsymbol{E}$ would do at $\boldsymbol{E}$'s turn. Instead of focusing on relations, it would also be possible to analyze Backward Induction in terms of strategy profiles in equilibrium, using the models of games in Chapters 6 and 12. We will not explore this alternative logical route in this book. ■

2.3 Defining Backward Induction in fixed point logics

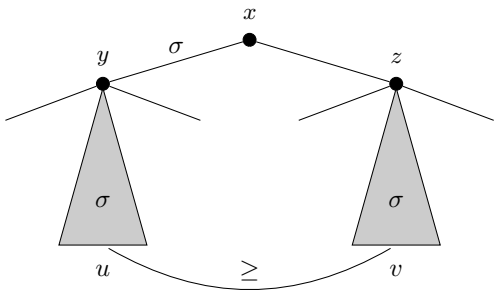
Defining the Backward Induction solution is a benchmark for logics of games.[10] We start by citing a result from van Benthem et al. (2006b).

10 The game-theoretic literature distinguishes the Backward Induction path: the actual history produced, and the Backward Induction strategy that includes off-path behavior whose interpretation may be questioned. The characterizations of Aumann (1995, 1999) are for the Backward Induction path only. Our approach covers the whole strategy.

THEOREM 2.1 On finite extensive games, the Backward Induction strategy is the largest subrelation σ of the total *move* relation that has at least one successor at each node, while satisfying the following property for all players i:

> *Rationality* (RAT) No alternative move for the current player i yields outcomes via further play with σ that are all strictly better for i than all outcomes resulting from starting at the current move and then playing σ all the way down the tree.

The following picture illustrates more concretely what this says:



The shaded area in this diagram is the part that can be reached via further play with our strategy σ. The proof that the property RAT is necessary and sufficient for capturing the Backward Induction solution is by a straightforward induction on the depth of finite game trees.

Stated in more syntactic terms, RAT expresses a "confluence property" for action and preference, where steps in different subtrees get linked by preference:

$$CF \quad \forall x \forall y \Big(\big(Turn_i(x) \wedge x \sigma y \big) \rightarrow \forall z \big(x \, move \, z \rightarrow \exists u \exists v (end(u) \wedge end(v) \wedge y \sigma^* v \wedge z \sigma^* u \wedge u \leq_i v) \big) \Big)$$

This is the basis for definability of the Backward Induction solution in a well-known system from computational logic, *first-order fixed point logic* LFP(FO) (Ebbinghaus & Flum 1999). The system LFP(FO) extends first-order logic with

smallest and greatest fixed point operators producing inductively defined new predicates of any arity, a highly expressive device going far beyond the μ-calculus of Chapter 1 that only defined new unary predicates.[11]

THEOREM 2.2 The Backward Induction relation is definable in LFP(FO).

Proof In the consequent of the above $\forall\forall\exists\exists$ format for CF (confluence), all occurrences of the symbol σ are syntactically positive. This positive syntax allows for a greatest fixed point operator defining *bi* in LFP(FO) as the following relation S:

$$\nu S, xy \bullet \forall x \forall y \Big(\big(\mathit{Turn}_i(x) \wedge Sxy \big) \rightarrow \forall z \big(x \mathit{move} z \rightarrow \exists u \exists v (\mathit{end}(u) \wedge \mathit{end}(v) \wedge y S^* v \wedge z S^* u \wedge u \leq_i v) \big) \Big)$$

It is shown in detail in van Benthem & Gheerbrant (2010) how the successive steps of the cautious Backward Induction algorithm match the approximation stages for the greatest fixed point denoted by this formula, starting from the total *move* relation, and refining it downward. ■

LFP(FO) is a fundamental system in its field, and hence our analysis has made a significant junction between game solution methods and logics of computation.

Variants Variants can be defined, too, with, say, an $\forall\forall\forall\exists$ format $\forall x \forall y ((\mathit{Turn}_i(x) \wedge x \sigma y) \rightarrow \forall z (x \mathit{move} z \rightarrow \forall u ((\mathit{end}(u) \wedge y \sigma^* u) \rightarrow \exists v (\mathit{end}(v) \wedge z \sigma^* v \wedge v \leq_i u))))$. This syntax is no longer positive for σ, and existence and uniqueness results now require an appeal to the well-foundedness of game trees, as well as the use of special logics for trees. These matters are studied extensively in Gheerbrant (2010).

2.4 Zooming out to modal logics of best action

Fixed point logics make solution procedures for games fully explicit in their notation. But logical description of behavior can take place at various levels, either zooming in on formal details that lie below a natural reasoning practice, or doing precisely the opposite, zooming out to useful abstractions that lie above the surface level. In the latter vein, often, we want to hide the details of a computation, and

11 The price for this expressive power is that validity in LFP(FO) is non-axiomatizable, and indeed of very high complexity. Still, this system has many uses, for instance, in finite model theory (Ebbinghaus & Flum 1999, Libkin 2004).

merely record properties of the notion of *best action*. An agent may just want to know what to do, without being bothered with all the details behind the relevant recommendation.

For this purpose, it would be good to extract a simple surface logic for reasoning with the ideas in this chapter, while hiding most of the earlier machinery. At this level of grain, modal preference logics become a good alternative again, on top of a standard logic of action as in Chapter 1. In particular, a new modality $\langle best \rangle$ now talks about the best actions that agents have available, encoding some particular style of recommendation. For concreteness, we will interpret it as follows:

> $\langle best \rangle \varphi$ says that is φ true in some successor of the current node that can be reached in one step via the bi relation.

THEOREM 2.3 The Backward Induction strategy is the unique relation σ satisfying the following modal axiom for all players i, and for all propositions p, viewed as sets of nodes:

$$(turn_i \wedge \langle best \rangle [best^*](\textbf{\textit{end}} \rightarrow p)) \;\rightarrow\; [move\text{-}i]\langle best^* \rangle (\textbf{\textit{end}} \wedge \langle pref_i \rangle p).$$

Proof The proof is a modal frame correspondence, applied to the earlier confluence property CF, which can be computed by standard modal methods.[12] ■

It is easy to find further valid principles in this language, expressing, for instance, that best actions are actions. The following natural issue in this setting goes back to van Otterloo (2005):

OPEN PROBLEM Axiomatize the modal logic of finite game trees with a *move* relation and its transitive closure, turn predicates and preference relations for players, plus a new relation *best* as computed by Backward Induction.[13]

This is just one instance of a global logic for practical reasoning that can be extracted from more detailed game structure. Section 2.7 will mention others.

---

12 The overall form here is a Geach-style convergence axiom (see van Benthem et al. 2012 for the latest results in frame correspondence for modal fixed point logics).

13 We get at least the basic modal logics for moves and for preference as discussed earlier, while the above bridge axiom fixes relevant connections between them. However, looking at details of this calculus, in Chapter 9, we also find a need for relativized predicates $best^P$ referring to moves that are best with a Backward Induction computation restricted to the submodel of all worlds satisfying P.

Pitfalls of complexity Global logics of best action may also be interesting from an unexpected computational perspective. You may think that surface logics should be easy, but there is a snag. The rationality in the above confluence picture entangles two binary relations in tree models for games: one for actions, and another for preference. The resulting *grid structure* for two relations on game trees can encode complex geometric "tiling problems," making the bimodal logics undecidable and nonaxiomatizable (cf. Harel 1985, van Benthem 2010b). We will elaborate on this phenomenon in Chapter 3 when discussing information processing agents with perfect memory (cf. Halpern & Vardi 1989), and once more in Chapter 12 on logics for strategic games with grid-like matrix structures. There is an interesting tension here between two views of simplicity. On the one hand, rationality is an appealing property guaranteeing uniform predictable behavior of agents, but on the other hand, rationality may have a high computational cost in the complexity of its induced logic of agency.[14]

2.5 Conclusion

The main points In this chapter, we have seen how games support a natural combination of existing logics for action and preference. We showed how the resulting game logics with action and preference can deal with the preference structure that is characteristic for real games, up to the point of defining the standard benchmark of the Backward Induction solution procedure. These logics came in two natural varieties of detail. Zooming out on basic global patterns, we found a modal logic of best action that seems of general interest in practical reasoning. Zooming in on details of solution procedures, we showed how the Backward Induction strategy can be defined in richer fixed point languages for games, in particular, the logic LFP(FO) for inductive definitions. This reinforces the general point made in Chapter 1 that game-theoretic equilibrium notions match up well with fixed point logics. In this way, our analysis creates a junction between game theory, computational logic, and philosophical logic. This combination of strands will continue as this book proceeds.

Open problems The themes of this chapter also suggest a number of open problems. These include axiomatizing modal logics of best action, and exploring the

14 On the computational complexity of game solution procedures as analyzed in this chapter, cf. Szymanik (2013).

computational complexity of logics of action and preference, especially the effects of varying bridge principles between the two components, of which rationality was just one. Of fundamental importance also would be finding the right structural equivalence for games with preferences, continuing a major theme from Chapter 1. For instance, should we now identify processes when only their best actions can be simulated? Further issues include defining strategies explicitly: in Chapter 1, programs from dynamic logic served this purpose; how can these be extended to deal with preference? Finally, there is the challenge of an extension to infinite games, where Backward Induction has a problematic status (Löwe 2003). Fixed point logics fit well with infinite models, and greatest fixed points like the one we found denote co-inductive objects such as never-ending strategies (cf. Chapter 18). However, the precise relation is still unclear in our setting.

Some of the issues mentioned here will be discussed in our final Section 2.7 on further directions, which can be skipped without loss of continuity. Also, a number of relevant considerations will return later on in this book, in Chapter 4 on strategies and in Chapter 8 on dynamic logics for game solution procedures.

2.6 Literature

This chapter is based on van Benthem (2002a), van Benthem et al. (2006b), and especially, van Benthem & Gheerbrant (2010).

Further texts with many additional insights on preference, games, and logic are Dégremont (2010), Gheerbrant (2010), Liu (2011), and Zvesper (2010).

2.7 Further directions

For the reader who wants more food for thought, we list a few further directions.

Entangling preference and belief Our first topic is an important reinterpretation of what we have done in this chapter. Looking more closely at the earlier confluence property, we see that it makes comparisons between current moves based on an assumption about future play, viz. that Backward Induction will be played henceforth. This reveals another aspect of game solution: it entangles preference with belief. Backward Induction, and indeed also other game solution methods, are really procedures for *creating expectations* of players about optimal behavior.

In line with this, our earlier version of rationality can be amended to a perhaps more sophisticated notion of "rationality-in-beliefs": Players play no move whose results they believe to be worse than those of some other available move. Beliefs will be the topic of later chapters, especially in Part II where we analyze the dynamics of game solution procedures as a process of successive belief changes. As will be shown in Chapter 8, strategies are very much like beliefs, in a precise formal sense.

Other social scenarios Backward Induction is just one scenario, and many other scenarios in games, or social settings generally, can be studied in the above spirit.

Variety of logics for games It is not written in stone that a language of games has to be a basic modal one. Stronger formalisms are also illuminating, such as the use of temporal Until operators in van Benthem (2002a) to define preferences between nodes in game trees. Further extensions occur in defining Nash equilibria (cf. Chapter 12): these require intersections of relations, a program operation that goes beyond PDL. Going back to Chapter 1, such alternative logics may also suggest new notions of process equivalence for games.

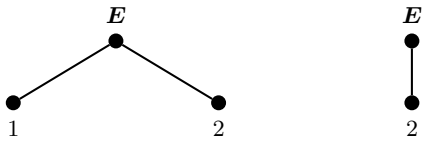
One more rough level: Deontic logic Our line of zooming out to best action has counterparts in recent top-level logics of complex social scenarios in terms of common sense notions such as "may" and "ought" that pervade ordinary discourse. Tamminga & Kooi (2008), Roy (2011), and Roy et al. (2012) provide further illustrations of this more global zoom level, relating games to deontic logics for reasoning about permissions and obligations.

Priority models for preference Our models for preference were sets of abstract worlds with a primitive betterness relation. This does not record why agents would have this preference, based on what considerations. Reasons for preference are explicitly present in the "priority graphs" of Liu (2011) that list the relevant properties of worlds with their relative importance. Betterness order on worlds is then derived from priority graphs in a natural lexicographical manner.

This richer style of analysis makes sense for games, too, since we often judge outcomes in terms of ordered goals that we want to see satisfied, rather than an immediate preference ordering. See Chapter 22 on some uses of players' goals in knowledge games over epistemic models, and Grossi & Turrini (2012) and Liu (2012) for an application of priority graphs to short-horizon games. Osherson & Weinstein (2012) present another take on reason-based preference, with various modal connectives reflecting reasons to desire that a sentence be true.

Game equivalence with preferences As in Chapter 1, a greater variety of languages also suggests variety in structural notions of game equivalence. Our earlier notions of game invariance become more delicate in the presence of preferences.

As a simple illustration, we can intuitively identify the following two non-bisimilar one-player games as the same:



Three approaches to equivalence have been explored in van Benthem (2002a): (i) direct simulation on preference links, (ii) pruning games to their best actions, and then using the standard action invariances of Chapter 1, and (iii) preference-equivalence of games as supporting the same Nash equilibria.

What complicates intuition here is an issue of equivalence for whom? The above games only seem equivalent for rational players $\boldsymbol{E}$, not for erratic or stupid ones. But then, a more general point arises. Should our recurrent question of game equivalence perhaps be relativized to "agent types" for players? Such an agent orientation would be a significant shift in perspective compared to standard process theories in computational logic, of a kind discussed further in Part II of this book.

4
Making Strategies Explicit

4.1 Strategies as first-class citizens

As we saw in our Introduction, much of game theory is about the existence of strategic equilibria. In the same spirit, many game logics have existential quantifiers saying that players have a strategy for achieving some purpose (cf. Parikh 1985, Alur et al. 2002, and Chapter 19), much like the power modalities discussed in earlier chapters. But the strategies themselves are not part of the formal language, and this leaves out a protagonist in the story of interaction. Strategies are what drives rational agency over time, unfolding via successive interactive moves. The widespread tendency in logic toward hiding information under existential quantifiers has been called ∃-sickness in van Benthem (1999).[23] To cure this, it makes sense to move strategies explicitly into our logics, and reason about their behavior, as one would do with plans. There are several ways of doing this. One is to use logics of programs to deal with strategies, and we will show how this works with propositional dynamic logic PDL. Another approach is by inspecting the elementary reasoning about strategies underlying basic game-theoretic results, and designing an abstract calculus of strategies that can represent these naturally. In particular, this emancipation makes sense for the logic games of Part IV of this book, where hiding strategies under existential power quantifiers is common (cf. Chapter 25).

23 Other instances of ∃-sickness are doing logic of provability rather than of proofs, or of knowability rather than of knowing. In general, suffixes such as -ility should be red flags!

The logic of strategies is a recent area, and we will not present any definitive version, but the theme will return in Chapters 5 and 18 and in Part V.

4.2 Defining strategies in dynamic logic

Our first approach to strategies treats them as programs in a well-known logic.

Dynamic logic of programs Let us recall the basics of propositional dynamic logic, PDL, introduced in Chapter 1. PDL was designed originally to study imperative computer programs, or complex actions, that use the standard operations of sequential composition (;), guarded choice (IF... THEN... ELSE...), and guarded iteration (WHILE... DO...).

DEFINITION 4.1 Propositional dynamic logic
The language of PDL defines formulas and programs in a mutual recursion, with *formulas* denoting sets of states (i.e., they are local conditions on states of a process), while *programs* denote binary transition relations between states, consisting of ordered pairs (input state, output state) for the successful executions. Programs are built from atomic actions (moves) $a, b, \ldots$, and tests $?\varphi$ for all formulas φ, using the three "regular operations" of ; (sequential composition), $\cup$ (nondeterministic choice) and * (nondeterministic finite iteration). Formulas are constructed from atoms and Booleans as in a basic modal language, but now with dynamic modalities $[\pi]\varphi$ interpreted as follows in the process models $\boldsymbol{M}$ of Chapter 1

$$\boldsymbol{M}, s \models [\pi]\varphi \quad \text{iff} \quad \varphi \text{ is true after every successful execution of } \pi \text{ starting at } s.$$

This is the standard way of describing effects of programs or actions. ■

This system applies directly to games. In Chapter 1, the total *move* relation was a union of atomic relations, and the modal pattern for the existence of a winning strategy was $[a \cup b]\langle c \cup d\rangle p$. PDL focuses on finitely terminating programs, a restriction that one can question in general (see Chapter 5 on strategies in infinite games). However, it makes good sense in finite games, or with infinite strategies whose local steps are finite terminating programs. For safety's sake, we will work with finite games in this chapter, unless explicitly stated otherwise.

As for a calculus of reasoning about complex actions, the valid laws of PDL are decidable, and it has a complete set of axioms analyzing the regular program constructions in a perspicuous way (Harel et al. 2000, van Benthem 2010b).

Strategies defined by programs Strategies in game theory are partial functions on players' turns, given by instructions of the form "if my opponent plays this, then I play that." But as we have seen in Chapters 1 and 2, more general strategies are transition relations with more than one best move. They are like plans that can be useful by just constraining moves, without fixing a unique course of action. Thus, on top of the hard-wired *move* relation in a game, we now get new defined relations, corresponding to players' strategies, and these can often be defined explicitly in a PDL program format.[24]

As an example, the earlier "forcing modality" can be made explicit as follows.

FACT 4.1 For any game program expression σ, PDL can define an explicit forcing modality $\{\sigma, i\}\varphi$ stating that σ is a strategy for player i forcing the game, against any play of the others, to pass only through states satisfying φ.

Proof The formula $[((?\boldsymbol{turn_E}\,;\,\sigma)\,\cup\,(?\boldsymbol{turn_A}\,;\,\mathit{move}\text{-}\boldsymbol{A}))^*]\varphi$ defines the forcing. This says that following the program σ at $\boldsymbol{E}$'s turns and any move at $\boldsymbol{A}$'s turns always yields φ-states. ■

A related observation is that, given definable relational strategies for players $\boldsymbol{A}$ and $\boldsymbol{E}$, we get to an outcome for the game that can be defined as well.

FACT 4.2 Outcomes of running joint strategies σ, τ can be described in PDL.

Proof The formula $[((?\boldsymbol{turn_E}\,;\,\sigma)\,\cup\,(?\boldsymbol{turn_A}\,;\,\tau))^*](\boldsymbol{end}\to\varphi)$ does the job. ■

Flat strategy programs Our program format has some overkill. A strategy prescribes one move at a time, subject to local conditions. Then, local iterations * make little sense, and one can restrict attention to PDL programs using only atomic actions, tests, ;, and $\cup$. These can easily be brought into a normal form consisting of a union of "guarded actions" of the form

$$?\varphi_1\,;\,a_1\,;\,\cdots\,;\,?\varphi_n\,;\,a_n\,;\,?\psi$$

This makes a strategy a set of conditional rules that are applicable only under local conditions and with specified postconditions.[25]

---

24 It seems reasonable to require that relations for strategies must be non-empty on turns of the relevant player. All that we have to say is compatible with this.

25 This format for strategies of belief revision and model change was proposed in van Benthem & Liu (2007), and more general versions occur elsewhere (van Eijck 2008, Girard et al. 2012, Ramanujam & Simon 2008).

Even this may still be too much, since prescribing sequences of actions $a_1 ; \cdots ; a_n$ only makes sense when a player has several consecutive turns, while most games have alternating scheduling. "Flat programs" often suffice, being unions of guarded actions of the form $?\varphi ; a ; ?\psi$.

Expressive completeness On a model-by-model basis, the expressive power of PDL is high (Rodenhäuser 2001). Consider any finite game $\boldsymbol{M}$ with strategy σ for player i. As a relation, σ is a finite set of ordered pairs (s, t). In case we have an "expressive model" $\boldsymbol{M}$ whose states s are definable in our modal language by formulas def_s,[26] we can define pairs (s, t) by formulas $def_s ; a ; def_t$, where a is the relevant move, and take the relevant union (note that this is indeed a flat program in the earlier syntactic sense).

FACT 4.3 In expressive finite extensive games, all strategies are PDL-definable.

PDL and strategy combination PDL also describes combinations of strategies in terms of operations on relations (van Benthem 2002a). A basic operation is union, allowing all possible moves according to both strategies. Union merges two plans, constraining players' moves into a common weakening. The laws of PDL describe how this operation behaves in single steps:

$$\langle \sigma \cup \tau, i \rangle \varphi \leftrightarrow \langle \sigma, i \rangle \varphi \vee \langle \tau, i \rangle \varphi$$

It may be of more interest to look at strategy modalities $\{\sigma, \boldsymbol{E}\}\varphi$ defined as before with repeated steps. In that case, it is easy to see that distribution fails (see Chapters 11 and 19 for more on this). There are obvious counterexamples to:

$$\{\sigma \cup \tau, i\}\varphi \leftrightarrow \{\sigma, i\}\varphi \vee \{\tau, i\}\varphi$$

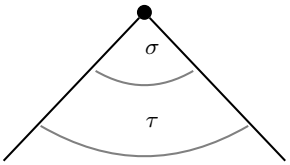
and we only have monotonicity laws such as:

$$\{\sigma, i\}\varphi \rightarrow \{\sigma \cup \tau, i\}\varphi$$

Perhaps a more important operation on relational strategies is the *intersection* $\sigma \cap \tau$, which combines two separate recommendations. In some cases, this may not leave any moves at all. But in general, intersection mimics an intuitive composition of strategies. Think of different players, with a strategy for one player allowing any

26 This can be achieved using temporal past modalities to describe the history up to s.

move for the other. Intersection then produces joint outcomes when both players play their given strategy. But we can also think of intersection as composing different partial strategies for the same player. Suppose that a strategy σ works up to a certain level in the tree, imposing no restrictions afterward, while strategy τ imposes no restrictions first, but kicks in after the domain of σ has come to an end:



In this case, the intersection has the effect of the intuitive composition $\sigma ; \tau$.

PDL has no reduction axiom for $\sigma \cap \tau$, not even for single steps, since intersection is not a regular program operation. But we do keep some general laws of strategy calculus, such as the validity of

$$(\{\sigma, i\}\varphi \wedge \{\tau, i\}\psi) \to \{\sigma \cap \tau, i\}(\varphi \wedge \psi)$$

Finding a complete axiomatization for forcing statements with union and intersection of strategies seems an open problem, but given the earlier PDL definition of forcing, an implicit calculus lies embedded inside PDL with intersection (cf. Harel et al. 2000). A recent, more in-depth study following up on our theme of strategy combination is van Eijck (2012).

Zooming out: Powers and explicit strategies The forcing modalities $\{\sigma, i\}\varphi$ used in this chapter so far have explicit strategies inside them. They may be viewed as explicit counterparts to the more implicit forcing modalities $\{i\}\varphi$ of Chapter 1 that quantified existentially over strategies for player i, zooming out on mere powers without the methods for achieving them. As we noted earlier, different zoom levels may have their own uses, and in particular, it makes sense to have both here.

Mere power modalities for players have an important use that is not subsumed by our explicit approach. If we want to say that a player *lacks a strategy* for achieving a certain goal, the following formula will do:

$$\neg\{i\}\varphi$$

No explicit version can express the same, since, in general, we cannot list all possible strategies σ in order to deny that they force φ. Thus, it makes sense to have logics that combine both implicit and explicit forcing statements.

A further and perhaps surprising use of programs plus existential forcing modalities occurs when defining strategies that make powers explicit. Given any player i and formula φ, define the PDL program

$$\sigma_{\varphi,i} = ?\boldsymbol{turn}_i\,;\, move\,;\, ?\{i\}\varphi$$

For instance, this is one way to define the success strategy in the Zermelo coloring algorithm of Chapter 1: "make sure that you keep going to winning positions." Now we can state a simple equivalence that follows easily from our earlier definitions.

FACT 4.4 $\{i\}\varphi \leftrightarrow \{\sigma_{\varphi,i}, i\}\varphi$ is valid.

Proof From right to left, if we have the specific strategy, then, a fortiori, the existential forcing modality holds. From left to right, we check that current truth of the forcing statement will persist along the stated strategy. This follows a recursion as in Chapter 1. (a) If it is i's turn, then there exists at least one successor where $\{i\}\varphi$ is true, (b) If it is j's turn, then all successors have this property, (c) If we are in an endpoint, then φ holds, and we are done. On finite games, this says that we can force total histories whose endpoints satisfy φ.[27] ■

On infinite games, the preceding equivalence still holds with the following understanding of forcing modalities: "the player can force a set of total histories at all of whose stages φ is true." The logic of this temporal forcing modality has some delicate aspects that we postpone until Chapter 5.

Details of definability What borderline separates the above trivializing rule "be successful" from strategies with real content? One crucial factor is *definability* of strategies in suitably restricted formalisms. For instance, while the above forcing modality of success looks at the whole future of the game, in practice, strategies often lack such forward-looking tests. Rather, they check for what happened in the past of the game tree, or test even just local assertions at the current node that require neither futurology nor history. Examples of strategies that are definable in such restricted ways, including memory-free ones that can be surprisingly powerful,

27 Variant conditions arise here by structuring φ. For instance, using suitable dynamic modalities, one can express that φ is true everywhere on the branches forced.

will be found throughout this book (cf. Chapters 18 and 20). Still, the picture is diverse. Some natural strategies seem to be intrinsically forward-looking, such as the Backward Induction strategy of Chapter 2. Programs in dynamic logic can define and calibrate some of these kinds of strategies, although the more general format for inspecting past and future would be the temporal languages to be discussed in the next chapter.

In summary, dynamic logic does a good job at defining strategies in simple extensive games. While we have also seen a need for richer computational logics serving similar purposes,[28] our point of existence for useful logical calculi of strategies has been made. Beyond analyzing strategic reasoning, such formalisms could also be used for calibrating strategies in hierarchies of definability, from simpler to more complex rules of behavior.

4.3 General calculus of strategies

While PDL programs can define strategies in specific games, this is done on a local game-by-game basis. And also when viewed as a generic proof calculus, we proposed this system mainly because of its track record for process graphs, not as a result of analyzing specific pieces of strategic reasoning. In other words, we started from systems at hand, not from an initial reasoning practice.

Another approach to designing strategy logics proceeds by inspecting standard game-theoretic arguments. We now present two examples of this approach to analyzing strategic reasoning, although we will not propose a final new calculus. Further examples of the same "quasi-empirical" approach will be found in Chapter 5 and in Part V of this book on logics of strategic powers in complex games.

Looking ahead, consider the logic of players' powers presented in Chapter 11.

EXAMPLE 4.1 Strategizing dynamic power logic
Consider a Boolean choice game $G \cup H$ where player $\boldsymbol{E}$ starts by choosing to play either game G or game H. In terms of the forcing notation of Chapter 1, now with games explicitly marked, the following principle is obviously valid:

$$\{G \cup H, \boldsymbol{E}\}\varphi \leftrightarrow \{G, \boldsymbol{E}\}\varphi \vee \{H, \boldsymbol{E}\}\varphi$$

28 A simpler view of strategies makes them finite sequences of basic actions by automata (cf. Ramanujam & Simon 2008). Such strategies have also been added to temporal logics of games (cf. Broersen 2009, Herzig & Lorini 2010).

The intuitive reasoning is as follows. From left to right, if $\boldsymbol{E}$ has a strategy forcing outcomes satisfying φ in $G \cup H$, then the first step in that strategy describes $\boldsymbol{E}$'s choice, left or right, and the rest of the strategy gives outcomes with φ in the chosen game. And vice versa, say, if $\boldsymbol{E}$ has a strategy forcing φ in game G, then prefixing that strategy with a move of going left gives a strategy forcing φ in $G \cup H$. ■

Right under the surface of this example lies a calculus of arbitrary strategies σ. Our first argument introduced two operations: $head(\sigma)$ takes the first move of the strategy, and $tail(\sigma)$ gives the remainder. Clearly, there are natural laws governing these operations, in particular:

$$\sigma = \big(head(\sigma), tail(\sigma)\big)$$

Next, the converse argument talks about prefixing an action a to a strategy σ, yielding a concatenated strategy $a\,;\,\sigma$ equally satisfying natural laws such as:

$$head(a\,;\,\sigma) = a \qquad\qquad tail(a\,;\,\sigma) = \sigma$$

This suggests that there is a basic strategy calculus underneath our common reasoning about games.[29] The following example takes its exploration a bit further.

EXAMPLE 4.2 Exploring basic strategy calculus
Consider the following simple sequent derivation for a propositional validity:

$$\dfrac{\dfrac{\dfrac{A \Rightarrow A}{A, B \Rightarrow A} \qquad \dfrac{B \Rightarrow B}{A, B \Rightarrow B}}{A, B \Rightarrow A \wedge B}}{\dfrac{A, B \Rightarrow (A \wedge B) \vee C}{A, B \vee C \Rightarrow (A \wedge B) \vee C}} \qquad \dfrac{\dfrac{C \Rightarrow C}{A, C \Rightarrow C}}{A, C \Rightarrow (A \wedge B) \vee C}$$

$$\dfrac{A, B \vee C \Rightarrow (A \wedge B) \vee C}{A \wedge (B \vee C) \Rightarrow (A \wedge B) \vee C}$$

Here is a corresponding richer form indicating strategies:

29 Interestingly, the head and tail operations suggest a co-inductive view (cf. Venema 2006) where strategies are observed and then return to serving mode, rather than the inductive construction view of PDL programs. Chapters 5 and 18 have more on this.

$$x : A \Rightarrow x : A \qquad y : B \Rightarrow y : B$$
$$x : A, y : B \Rightarrow x : A \qquad x : A, y : B \Rightarrow y : B \qquad z : C \Rightarrow z : C$$
$$x : A, y : B \Rightarrow (x, y) : A \wedge B \qquad x : A, z : C \Rightarrow z : C$$
$$x : A, y : B \Rightarrow \langle l, (x, y)\rangle : (A \wedge B) \vee C \qquad x : A, z : C \Rightarrow \langle r, z\rangle : (A \wedge B) \vee C$$
$$x : A, u : (B \vee C) \Rightarrow IF\, head(u) = l\, THEN\, \langle x, tail(u)\rangle\, ELSE\, tail(u) : (A \wedge B) \vee C$$
$$v : A \wedge (B \vee C) \Rightarrow IF\, head((v)_2) = l\, THEN\, \langle (v)_1, tail((v)_2)\rangle\, ELSE\, tail((v)_2) : (A \wedge B) \vee C$$

The latter format can be read as proof construction, but we can also read what we have produced as a construction of strategies.[30] Its key operations are

storing strategies for a player who is not to move	$\langle\ ,\ \rangle$
using a strategy from a list	$(\ )_i$
executing the first action of a strategy	$head(\ )$
executing the remaining strategy	$tail(\ )$
making a choice dependent on some information	IF THEN ELSE

Clearly, this repertoire is different from PDL, and it may also have a quite different logical rationale, with a calculus more like those found in type theories.[31] A matching general strategy calculus might manipulate statements of the form "σ is a strategy forcing outcomes satisfying φ for player i in game G." ■

We will return to the logical structure of concrete instances of strategic reasoning at various places in this book, starting in Chapter 5 on infinite games.

4.4 Strategies in the presence of knowledge

Strategies also work in more complex settings, such as the imperfect information games of Chapter 3. Interesting new issues arise in this setting. For a start, in this case, the important strategies were the uniform ones, prescribing the same move at positions that a player cannot distinguish, or equivalently, making the action

30 For instance, in addition to being about proofs, taking a leaf from Chapter 14 of this book, the given proof is also a recipe for turning any winning strategy of the verifier in an evaluation game for $A \wedge (B \vee C)$ into a winning strategy for the game $(A \wedge B) \vee C$.

31 One analogy are type-theoretic statements $t : P$ saying that t is a proof of proposition P, or an object with property P. Type theories have rules encoding constructions of complex types (Barendregt 2001). See Abramsky & Jagadeesan (1994) for systematic links with games and strategies.

chosen dependent only on what a player knows. As in Section 4.2, we can add PDL-style programs to this setting. But there is a twist.

Knowledge programs It makes sense to impose a restriction now to the "knowledge programs" of Fagin et al. (1995), whose only test conditions for actions are knowledge formulas of the form $K\varphi$. More precisely, in our setting, we want knowledge programs that can serve as executable plans for an agent i, whose test conditions φ must have the property that i always knows whether φ is the case. Without loss of generality, this means that we can restrict to test conditions of the form $K_i\varphi$, since the following equivalence is valid:

$$U(K_i\varphi \vee K_i\neg\varphi) \to U(\varphi \leftrightarrow K_i\varphi)$$

As a special case, we take *flat knowledge programs*, with the special format defined earlier for flat PDL programs, with only one-step moves. These seem close to uniform strategies, but how close, precisely? First, we need a generalization. Uniform strategies as defined in Chapter 4 were functional, but our PDL programs are relational. Let us say that a relational strategy σ for player i is *uniform* if, whenever $\boldsymbol{turn}_i x$, $\boldsymbol{turn}_i y$, and $x \sim_i y$, then σ allows the same actions at x and y.

This is related to a general epistemic requirement on game models that occurs in the literature (cf. Osborne & Rubinstein 1994), namely, that players should know all actions available to them.[32] In modal terms, this imposes the condition:

$$(\boldsymbol{turn}_i \wedge \langle a\rangle\top) \to K_i\langle a\rangle\top, \text{ or even } (\boldsymbol{turn}_i \wedge \langle a\rangle\top) \to C_{\{i,j\}}\langle a\rangle\top$$

Making this assumption, the following implication becomes valid.

FACT 4.5 Flat knowledge programs define uniform strategies.

Proof Knowledge conditions have the same truth value at epistemically indistinguishable nodes for a player, and by the assumption, the available moves are the same. Hence the transitions defined by the program are the same as well. ■

The following result states a converse, given some conditions on the power of models for defining nodes such as those in Section 4.2 (van Benthem 2001b).[33]

32 In other words, players get no new knowledge from inspecting available moves.

33 Here we assume that models are bisimulation-contracted for action and uncertainty links. This ensures that each epistemic equivalence class will have a unique modal definition in our language, by the results of Chapter 1.

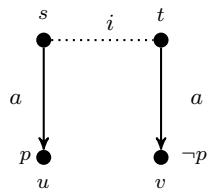
FACT 4.6 On expressive finite games of imperfect information, the uniform strategies are definable by knowledge programs in epistemic PDL.

Proof One enumerates the action of the given uniform strategy as we did for PDL programs in the non-epistemic case. We only need to say what the strategy allows on each epistemic equivalence class for the relevant player. This can be stated using the modal definitions of the equivalence classes, while noting that these definitions are known, making them suitable as test conditions for a knowledge program. ■

Further entanglements of knowledge and action The merge of epistemic logic with logics of strategies operates at two levels (see van Benthem 2012e for more on what follows). One level is propositional: preconditions and postconditions of actions can now be epistemic. We saw epistemic preconditions in knowledge programs, and Chapter 22 will give examples of epistemic postconditions in knowledge games, such as being the first to know a certain secret. The second level is dynamic, since actions can now be "epistemized" in different ways. Knowledge programs showed how test conditions can be epistemic. Further, actions can be epistemic by themselves, such as making an observation or asking a question. This special sort of epistemic action will be a main focus in Chapter 7 and beyond.

The two levels, propositional and dynamic, interact. For instance, one natural issue that arises is to which extent agents know the effects of their actions. For instance, should there be introspection given the epistemic nature of knowledge programs? Suppose that a knowledge program π for player i guarantees effect φ in a model, that is, the forcing statement $\{\pi, i\}\varphi$ holds everywhere, does the player know this? Actually, it is easy to see that this can fail.

EXAMPLE 4.3 Not knowing what you are doing
Let a single player have two indistinguishable worlds s and t, and only one action a at each, leading to worlds u and v where u has the property p, while v does not.



In this model, introspection fails: $?\top\,;a$ is a knowledge program for which $[?\top\,;a]p$ is true at s, but not at t, whence $K[?\top\,;a]p$ is false at s. ■

However, there is another issue of entanglement here. Suppose that we do know the effects of our basic actions, then what about complex actions defined by knowledge programs? Here the axioms of PDL have interesting things to say. For instance, suppose that we know all the effects of some program π_1. Since those effects may include statements $[\pi_2]\varphi$ for other programs π_2, it follows that we know $[\pi_1][\pi_2]\varphi$, and hence all the effects of longer programs $\pi_1 ; \pi_2$.

A final issue of entanglement has to do with our notion of perfect recall from Chapter 3. It might be called "epistemic grip": even if players do know the effect of a program, what will they know at intermediate stages of following a strategy? Consider the problem of following a guide in a bog, but having forgotten why we trusted the guide in the first place. In imperfect information games, if a player knows at the start that $\{\pi, i\}\varphi$ is the case, does it follow that, at each later stage, the player knows that the remaining strategy played yields outcomes with φ? Here recall the commutation of knowledge and action for players with perfect memory:

$$K_i[a]\varphi \to [a]K_i\varphi$$

FACT 4.7 If players have perfect recall for atomic actions, then they have it for all complex knowledge programs.

Proof To see this, it suffices to look at the following strings of implications, with the relevant inductive hypothesis presupposed at various steps:

(a) $K[\pi_1 ; \pi_2]\varphi \to K[\pi_1][\pi_2]\varphi \to [\pi_1]K[\pi_2]\varphi \to [\pi_1][\pi_2]K\varphi \to [\pi_1 ; \pi_2]K\varphi$

(b) $K[\pi_1 \cup \pi_2]\varphi \to K([\pi_1]\varphi \wedge [\pi_2]\varphi) \to (K[\pi_1]\varphi \wedge K[\pi_2]\varphi) \to$
$([\pi_1]K\varphi \wedge [\pi_2]K\varphi) \to [\pi_1 \cup \pi_2]K\varphi$

These show how intermediate knowledge proceeds. The crucial step now is that of tests. For arbitrary tests, $K[?\alpha]\varphi$ implies $K(\alpha \to \varphi)$, but there is no guarantee at all that this implies $\alpha \to K\varphi$. However, this is different for knowledge tests:

$K[?K\alpha]\varphi \to K(K\alpha \to \varphi) \to (K\alpha \to K\varphi)$ is valid in epistemic $S4$. ■

These are just a few of the issues connecting knowledge and action in games.[34]

34 In imperfect information games for players with perfect recall, all uniform strategies lead to epistemic grip (van Benthem 2001b). The converse question seems open.

Know-how and understanding strategies Programs are also interesting as epistemic objects in their own right, since they represent the intuitive notion of knowing how as opposed to merely knowing that. There is an extensive literature on what know-how is, but the game setting makes the issue quite concrete.

Again, the entanglement of the two notions is of particular interest. In addition to knowing how involving knowing that, there is also the fundamental notion of "knowing a plan" that seems crucial to rational agency. There is no generally accepted explication of what this means. One line is to ask, as we did in the above, for propositional knowledge about the effects that a plan will achieve.[35] However, intuitively, more is involved in knowing a plan. Consider what we want genuine learning to achieve: not just correct propositional knowledge, but also the ability to engage in a practice based on the plan. In education, we teach know-how at least as much as know-that, but what is it, really?

The contrast may be highlighted in terms of understanding a strategy or a plan versus merely knowing it. In addition to propositional knowledge of effects of a plan, or parts of it, there are other key features such as "robustness": that is, counterfactually knowing the effects of a plan under changed circumstances, or the ability to modify it as needed. And there are further tests, such as a talent for zoom: that is, being able to describe a plan at different levels of detail, moving up or down between grain levels as needed.[36]

4.5 Conclusion

The main points Strategies are so important to games and to social agency in general that they deserve explicit attention as objects of study. We have shown how to do this, by adding explicit strategies to existing logics of games. First we showed how the programs of propositional dynamic logic can be used for defining strategies and reasoning about them. Next, changing tacks, we showed how a more general

35 This issue also plays in the area of epistemic planning (Bolander & Andersen 2011, Andersen et al. 2012), where different kinds of knowledge or beliefs are important: about where we are in following some current plan, but also about how we expect the process to develop over time.

36 Similar issues arise in analyzing what it means for someone to understand a formal proof, and useful intuitions might be drawn from mathematical practice.

core calculus of strategic reasoning can be extracted from basic game-theoretic arguments. Finally, we showed how strategy logics mix well with epistemic structure, proving a number of results about imperfect information games, and raising some issues of better understanding know-how as opposed to know-that.

Open problems No established strategy logic exists comparable to game logic. Thus, finding general core languages for defining strategies and matching calculi for strategic reasoning seems a natural goal. As we have suggested, this may require fieldwork in analyzing good benchmarks. As we will see with various examples in Chapters 5 and 25, there are at least two sources for this in the present book. One is the logic games of Part IV that provide many instances of basic strategic reasoning. Another source are key strategies in game theory, including simple widespread rules such as Tit for Tat, discussed in our Introduction. We will continue with this exploration in Chapter 5.

Additional open problems arise with understanding the interplay of strategies with knowledge and information change, as discussed above, including a better understanding of knowing how. Beyond that, we need to understand the entanglement with other epistemic attitudes that are crucial to action, such as belief, and with acts of belief revision.

The next obvious desideratum in the context of this book is extending our analysis, local or generic, to games with preference structure. Defining the Backward Induction strategy of Chapter 2 in a basic logic of strategies would be an obvious benchmark. Finally, making strategies, which are complex actions, into first-class citizens, fits well with the logical dynamics program of Part II of this book, but an optimal integration has not happened yet.

4.6 Literature

This chapter is based on van Benthem (2012a). Some parts have also been taken from van Benthem (2012e) and van Benthem (2013). This program for putting strategies in the limelight was presented originally at a half-year project at the Netherlands Institute for Advanced Studies (NIAS) in 2007.

A useful anthology on logics of strategies from the 1990s is Bicchieri et al. (1999). In the meantime, many relevant publications have appeared, including work by the Chennai Group working with automata theory (cf. Ramanujam & Simon 2009, Ghosh & Ramanujam 2011), by the CWI group on strategies in PDL and related formalisms (cf. Dechesne et al. 2009, Wang 2010), the STRATMAS

Project (`http://www.ai.rug.nl/~sujata/documents.html`) on many different approaches to strategic reasoning, and on the borderline with the dynamic-epistemic logic of Part II of this book (Pacuit & Simon 2011).

4.7 Further directions

We have mentioned a number of open problems already in the main text of this chapter. Some further detail now follows.

Systems thinking versus a quasi-empirical approach As we have suggested at several places, there are at least two approaches to designing logics of strategies. One starts from general system considerations and analogies with notions such as programs or automata, and it is the dominant approach in this book. But one could also start by independently compiling a repertoire of basic strategies of wide sweep (as has been done for algorithms), plus the kind of basic reasoning that establishes their properties. Striking phenomena here are the ubiquity and power of simple identity strategies such as Tit for Tat in game theory, or copy-cat in computation, or indeed variable identification in logic itself. In particular, this suggests looking at definability of strategies in simple logical languages, although we are not aware of any established calibration hierarchy. Hand in hand with this would be logical analysis of proofs establishing key results about strategies in the literature, a working style that will return at a number of places in this book.

Extending the dynamic logic approach We have only shown PDL at work for strategies in games with sequential turns. But parallel action is a common feature in games. This requires endowing atomic actions with more structure, giving them components for each player, and perhaps also the environment, as in Fagin et al. (1995). The logics of Chapters 12 and 20 provide examples of how this may be done, adding fine structure to basic events in terms of control by the players. Another type of extension concerns effects of strategies. So far, we looked at either endpoints, or at all future stages. But there are other intuitive success notions, in terms of intermediate effects. Let $\{\sigma\}^*\varphi$ say that strategy σ guarantees reaching a "barrier" of φ-positions in the game (a set that intersects each maximal chain). This may be the optimal setting for composition of strategies. What happens to game logic when we add barrier modalities?

Modal fixed point logics Richer formalisms than PDL also make sense for strategies. In Chapter 1, we saw a move from PDL to the modal μ-calculus that defines

a much richer set of recursive notions, including talk about infinite computations. But there is a problem: the μ-calculus has no explicit programs. Still, many formulas suggest a match. For instance, "keep playing a" (the non-terminating program WHILE $\top$ DO a) is a witness to the infinite a-branches claimed to exist by a greatest fixed point formulas $\nu p \bullet \langle a \rangle p$. An explicit program version of the μ-calculus might be a useful calculus of strategies. Attempts so far have only focused on terminating programs (Hollenberg 1998), and perhaps a better paradigm are the μ-automata of Bradfield & Stirling (2006). In Chapter 18, we will return to this theme, including strategic reasoning in graph games (cf. Venema 2006). In Chapter 2, we even used the much stronger fixed point logic LFP(FO), where similar points apply. Even so, one might say that fixed point logics do give explicit dynamic information about strategies, since the fixed point operators themselves refer to a fixed computational approximation procedure close to strategy construction.

Strategies and invariants Our discussion of strategies in PDL, and in particular, that of the trivializing power strategy "nothing succeeds like success," also suggests a further perspective. Many good strategies consist in maintaining a suitable *invariant* throughout the course of a game. This is true for parlor games such as Nim, but also for many of the logic games to be discussed in Part IV of this book. Indeed, our modal forcing statements about future success can themselves be viewed as abstract invariants, and the same is true for logical formulas in many game languages. But invariants can also be other structures: in some sense, the epistemic-doxastic models to be discussed in later chapters serve as invariants recording some (but not all) memory of past behavior. This book has no systematic theory of invariants to offer, but several topics have some bearing on this, such as the discussion in Chapters 18 and 25 of games viewed simultaneously at different levels.

Temporal logics Another broad paradigm that supports strategy calculus is that of temporal logics. Recent proposals include "strategizing" alternating temporal logic ATL (Alur et al. 2002, Ågotnes et al. 2007) or epistemic ATEL (van der Hoek & Wooldridge 2003, van Otterloo 2005), and analyses of games in interpreted systems (Halpern 2003a) or situation calculus (Reiter 2001). Temporal logic might be a better focus for the study of strategies in infinite games than the systems in our chapter, and we will give a few illustrations in Chapter 5.

From concrete to generic strategies The strategies in the systems of this chapter were concrete objects defined inside specific games. But there is also another level of generic strategies that achieve their effects across all games of some appropriate type. Examples are simple but powerful copying strategies such as Tit for

Tat or copy-cat, that work across a wide range of games. Generic strategies lie behind the logics of the game-constructing operations in Part V of this book, based on dynamic logic and linear logic, and they will appear in Chapters 19, 20, and 21.

Yet other approaches There are yet other approaches for making strategies explicit, such as automata theory (Ramanujam 2008) or type theory (Jacobs 1999). It remains to be seen how these fit with the logics in this chapter, although the Appendix to Chapter 18 has some relevant discussion. The study of strategies may also profit from concrete instances in other fields, such as the area of planning mentioned already in Chapter 3 (cf. Moore 1985, Bolander & Andersen 2011). Another mathematical paradigm for concrete strategies linked to logical tasks is formal learning theory (Kelly 1996).

Adding knowledge and belief There is more to adding informational attitudes to PDL than we have shown so far. First, one can perform a more drastic epistemization than we did, not just on propositions, but also on programs, making transitions themselves objects that can have epistemic structure, in the line of event models in dynamic-epistemic logic (cf. Chapter 7).[37] It is possible to go further in other ways. Our view of understanding a strategy as knowing its effects under changed circumstances mirrors counterfactual views in philosophy that make knowledge of φ a true belief that tracks truth in the following sense: if the world had been slightly different, we would still have a correct belief or disbelief about φ (Nozick 1981). This suggests incorporating belief into the structure of strategies, and indeed, Chapter 8 will view strategies as encoding beliefs. This twist implies a radical perspective, since beliefs come with belief revision, an ability to correct ourselves when contradicted by the facts. We will return to strategy revision in one of the points below (see also Chapter 9).

Adding preferences An intuitive sense of strategic behavior is based on motives and goals. Indeed, action and preference were deeply entangled in our logic for Backward Induction in Chapter 2. What about explicit strategies in this richer realm? We need extensions of our earlier modal preference logics that can define benchmarks like the Backward Induction strategy, perhaps in the style of the deontic dynamic logics of van der Meyden (1996).

---

37 An epistemic variant of "arrow logic" is used for this purpose in van Benthem (2011d).

Strategies across changing situations So far, we have studied strategies in fixed situations, which can lead to local solutions. What happens when a strategy has a proven effect, but we change the game to a new one? We will address in detail the topic of strategies in changing games in Chapter 9, but briefly explore some points presently. Recall our earlier point about understanding a strategy. A good plan should still work when circumstances change; it will be robust under at least small changes. But many strategies fall apart under change. For instance, the Backward Induction strategy can shift wildly with addition or deletion of moves.[38] Two options arise here, "recomputation" and "repair." Should we compute a new plan in a changed game, or repair the old plan? We often start with repair, and only recompute when forced. Could there be a serious theory of plan revision? Can we say more precisely when gradual changes are sufficient, and when they fail?[39] The issues of strategy structure, definability and preservation behavior arising here are still to be addressed systematically. Even for PDL, we know of no model-theoretic preservation theorems that relate program behavior across different models.

Conceptual clarification A final difficulty is the proliferation of undefined terms in the field, such as plan, strategy, agent type, or protocol. All point to similar things, and discussions become confusing. It would be good to fix terminology (cf. van Benthem et al. 2013). Protocols might be general constraints on a process; strategies might be ways of using one's freedom within these constraints; and agent types could be reserved for repeatable styles of action that agents have available.

---

38 One might say that the flexibility we seek is already given in the standard notion of a game, where a strategy has to work under any eventuality. One could collect all relevant cases of change into one "supergame," asking for one strategy working there. But such a pre-encoding seems far removed from our ordinary understanding of plans. In Chapter 6 and also in Part II, we will opt for working with small models instead.

39 For a nice example of repairing programs, see Huth & Ryan (2004).