of A_0 and $\mathcal{T}$, and $d_0 \in A_0^{\mathcal{I}}$. For $d \in \Delta^{\mathcal{I}}$ and $i \leq \mathsf{rd}(A_0)$, set

$$\mathsf{tp}_i(d) = \{A \in \mathsf{Def}_i(\mathcal{T}) \mid d \in A^{\mathcal{I}}\}.$$

We use $\mathcal{I}$ to guide the nondeterministic decisions of the algorithm. To do this, it is convenient to pass an element $d \in \Delta^{\mathcal{I}}$ as a virtual fourth argument to the procedure recurse such that $d \in A^{\mathcal{I}}$ for all A in the first argument τ.

Initially, we guide the algorithm to guess $\mathsf{tp}_{\mathsf{rd}(A_0)}(d_0)$ as the set τ in $\mathcal{ALC}$-worlds. Now let recurse be called with arguments $(\tau, i, \mathcal{T}, d)$, and assume that the **for all** loop is processing $A \in \tau$ with $A \equiv \exists r.B \in \mathcal{T}$. Then $d \in A^{\mathcal{I}}$ and thus there is a $d' \in B^{\mathcal{I}}$ with $(d, d') \in r^{\mathcal{I}}$. We guide the algorithm to guess $\mathsf{tp}_{i-1}(d')$ as the set τ. It remains to show that, when guided in this way, the algorithm returns true. This boils down to showing that all the guessed sets τ are types, which is straightforward using the semantics. □

We have thus proved the following result.

Theorem 5.5. *In $\mathcal{ALC}$, concept satisfiability and subsumption with respect to acyclic TBoxes are in* PSPACE.

Lower Bound

We now prove that the PSPACE upper bound from Theorem 5.5 is optimal by showing that concept satisfiability in $\mathcal{ALC}$ is PSPACE-hard, even without TBoxes. This implies that concept satisfiability is PSPACE-complete, both without TBoxes and with acyclic TBoxes.

The most common way to prove hardness for a complexity class $\mathcal{C}$ is to find an appropriate problem P that is already known to be hard for $\mathcal{C}$ and then to exhibit a polynomial time reduction from P to the problem at hand. In our case, the problem P is related to a game played on formulas of propositional logic and known to be PSPACE-complete [SC79].

A *finite Boolean game* (FBG) is a triple $(\varphi, \Gamma_1, \Gamma_2)$ with φ a formula of propositional logic and $\Gamma_1 \uplus \Gamma_2$ a partition of the variables used in φ into two sets of identical cardinality. The game is played by two players. Intuitively, Player 1 controls the variables in Γ_1 and Player 2 controls the variables in Γ_2. The game proceeds in $n = |\Gamma_1 \uplus \Gamma_2|$ rounds, with the players alternating. We assume that the variables in Γ_1 and Γ_2 are ordered. Player 1 moves first by choosing a truth value for the first variable from Γ_1. In the next round, Player 2 chooses a truth value for the first variable from Γ_2. Next, it is again Player 1's turn, who assigns

a truth value to the second variable in Γ_1, and so on. After n rounds, Player 1 wins the game if the resulting truth assignment satisfies the formula φ; otherwise, Player 2 wins. The decision problem associated with this game is as follows: given a game $(\varphi, \Gamma_1, \Gamma_2)$, decide whether Player 1 has a *winning strategy*, i.e., whether he can force a win no matter what Player 2 does.

Before reducing FBGs to concept satisfiability in $\mathcal{ALC}$, we give a formal definition of winning strategies. Fix a game $G = (\varphi, \Gamma_1, \Gamma_2)$, and let $n = |\Gamma_1 \uplus \Gamma_2|$. We assume that $\Gamma_1 = \{p_1, p_3, \ldots, p_{n-1}\}$ and $\Gamma_2 = \{p_2, p_4, \ldots, p_n\}$. A *configuration* of G is a word $t \in \{0,1\}^i$, for some $i \leq n$. Intuitively, the kth symbol in this word assigns a truth value to the variable p_k. Thus, if the current configuration is t, then a truth value for $p_{|t|+1}$ is selected in the next round. This is done by Player 1 if $|t|$ is even and by Player 2 if $|t|$ is odd. The *initial configuration* is the empty word ε. A *winning strategy* for Player 1 in G is a finite node-labelled tree (V, E, ℓ) of depth n, where ℓ assigns to each node $v \in V$ a configuration $\ell(v)$. We say that a node $v \in V$ is of *depth* i if v is reachable from the root by travelling along i successor edges. Winning strategies are required to satisfy the following conditions:

- the root is labelled with the initial configuration;
- if v is a node of depth $i < n$ with i even and $\ell(v) = t$, then v has one successor v' with $\ell(v') \in \{t0, t1\}$;
- if v is a node of depth $i < n$ with i odd and $\ell(v) = t$, then v has two successors v' and v'' with $\ell(v') = t0$ and $\ell(v'') = t1$;
- if v is a node of depth n and $\ell(v) = t$, then t satisfies φ.

Consider the game $G = (\varphi, \{p_1, p_3\}, \{p_2, p_4\})$, with

$$\varphi = \big(\, \neg p_1 \rightarrow p_2 \,\big) \wedge \big(\, (p_1 \wedge p_2) \rightarrow (p_3 \vee p_4) \,\big) \wedge \big(\, \neg p_2 \rightarrow (p_4 \rightarrow \neg p_3) \,\big).$$

Figure 5.2 shows a winning strategy for Player 1 in G. Intuitively, a winning strategy tells Player 1 how to play the game, no matter what Player 2 does. For example, if the current game configuration is 10, then Player 1 can look into the strategy tree for the (unique!) node $v \in V$ with $\ell(v) = 10$ and at its (unique!) successor v'. It is labelled 100, which advises Player 1 to set the truth value of p_3 to 0.

To reduce the existence of winning strategies in FBGs to the satisfiability of $\mathcal{ALC}$ concepts, we transform a game $G = (\varphi, \Gamma_1, \Gamma_2)$ into an $\mathcal{ALC}$ concept C_G such that Player 1 has a winning strategy in G if and only if C_G is satisfiable. The idea is to craft C_G such that every model

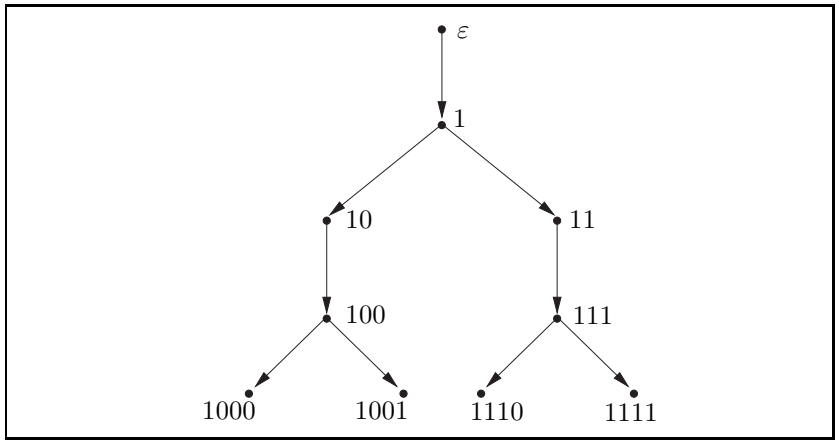

Fig. 5.2. A winning strategy for Player 1 in G.

of C_G describes a winning strategy for Player 1 in G and, vice versa, every such winning strategy gives rise to a model of C_G. The concept C_G uses a single role name r to represent the successor relation of the strategy tree. To describe the values of propositional variables, we use concept names $P_1, \dots, P_n$. Throughout this chapter, we use $C \to D$ as an abbreviation for $\neg C \sqcup D$, for better readability. Now, C_G is a conjunction whose conjuncts we define step by step, along with intuitive explanations of their meaning:

- For each node of odd depth i (i.e., Player 2 is to move), there are two successors, one for each possible truth value of p_{i+1}:

$$C_1 = \bigsqcap_{i \in \{1,3,\dots,n-1\}} \forall r^i.\big(\exists r.\neg P_{i+1} \sqcap \exists r.P_{i+1} \big),$$

  where $\forall r^i.C$ denotes the i-fold nesting $\forall r.\cdots\forall r.C$.
- For each node of even depth i (i.e., Player 1 is to move), there is one successor:

$$C_2 = \bigsqcap_{i \in \{0,2,\dots,n-2\}} \forall r^i.\exists r.\top.$$

  Note that, since P_{i+1} must be either true or false at the generated successor, a truth value for p_{i+1} is chosen "automatically".
- Once a truth value is chosen, it remains fixed:

$$C_3 = \bigsqcap_{1 \le i \le j < n} \forall r^j.\big((P_i \to \forall r.P_i) \sqcap (\neg P_i \to \forall r.\neg P_i) \big).$$

- At the leaves, the formula φ is true:

$$C_4 = \forall r^n.\varphi^*,$$

where φ^* denotes the result of converting φ into an $\mathcal{ALC}$ concept by replacing each p_i with P_i, $\sqcap$ with $\wedge$, and $\sqcup$ with $\vee$.

Now, we define $C_G = C_1 \sqcap \cdots \sqcap C_4$. It is easily verified that the length of C_G is quadratic in n, and that C_G can be constructed in time polynomial in n. The next lemma states that the reduction is correct.

Lemma 5.6. *Player 1 has a winning strategy in G if and only if C_G is satisfiable.*

Proof. (only if) Assume that Player 1 has a winning strategy (V, E, ℓ) with root $v_0 \in V$. We define an interpretation $\mathcal{I}$ by setting

- $\Delta^{\mathcal{I}} = V$,
- $r^{\mathcal{I}} = E$,
- $P_i^{\mathcal{I}} = \{v \in V \mid \ell(v) \text{ sets } p_i \text{ to } 1\}$ for $1 \leq i \leq n$.

We leave it as an exercise to verify that $v_0 \in C_G^{\mathcal{I}}$.

(if) Let $\mathcal{I}$ be a model of C_G, and let $d_0 \in C_G^{\mathcal{I}}$. We define a winning strategy (V, E, ℓ) with $V \subseteq \mathbb{N} \times \Delta^{\mathcal{I}}$. The construction will be such that

$(*)$ if $(i, d) \in V$, then d is reachable from d_0 in $\mathcal{I}$ by travelling i steps along r.

Start by setting $V = \{(0, d_0)\}$, $E = \emptyset$ and $\ell(0, d_0)$ to the initial configuration. We proceed in rounds $1, \ldots, n$. In each odd round i, iterate over all nodes $(i-1, d) \in V$ and do the following:

- select a $d' \in \Delta^{\mathcal{I}}$ such that $(d, d') \in r^{\mathcal{I}}$ (which exists since $d_0 \in C_2^{\mathcal{I}}$ and $(*)$ is satisfied by induction);
- add (i, d') to V, $((i-1, d), (i, d'))$ to E, and set $\ell(i, d') = tj$, where $t = \ell(i-1, d)$ and j is 1 if $d' \in P_i^{\mathcal{I}}$ and 0 otherwise.

In each even round i, iterate over all nodes $(i-1, d) \in V$ and do the following:

- select $d', d'' \in \Delta^{\mathcal{I}}$ such that $d' \in P_i^{\mathcal{I}}$, $d'' \notin P_i^{\mathcal{I}}$ and $\{(d, d'), (d, d'')\} \subseteq r^{\mathcal{I}}$ (which exist since $d_0 \in C_1^{\mathcal{I}}$ and $(*)$ is satisfied by induction);
- add (i, d') and (i, d'') to V, $((i-1, d), (i, d'))$ and $((i-1, d), (i, d''))$ to E, and set $\ell(i, d') = t1$ and $\ell(i, d'') = t0$, where $t = \ell(i-1, d)$.

Since $d_0 \in C_3^{\mathcal{I}}$ and $d_0 \in C_4^{\mathcal{I}}$, it is easy to prove that the resulting tree is a winning strategy for Player 1. □

We have thus established PSPACE-hardness of concept satisfiability in $\mathcal{ALC}$. Together with Theorem 5.5, we obtain the following result.

Theorem 5.7. *In $\mathcal{ALC}$, concept satisfiability and subsumption without TBoxes and with acyclic TBoxes are* PSPACE*-hard, thus* PSPACE*-complete.*

As in the case of $\mathcal{ALC}$, it is rather often the case that satisfiability without TBoxes and with acyclic TBoxes have the same complexity. However, there are also notable exceptions. One example is $\mathcal{ALC}$ extended with concrete domains. For some natural concrete domains, satisfiability without TBoxes is PSPACE-complete, and with respect to acyclic TBoxes it is NEXPTIME-complete.

5.1.2 General TBoxes

The aim of this section is to show that, in $\mathcal{ALC}$, the transition from acyclic TBoxes to general TBoxes increases the computational complexity from PSPACE to EXPTIME.

Upper Bound

We prove an EXPTIME upper bound for satisfiability with respect to general $\mathcal{ALC}$ concepts using a so-called *type elimination* algorithm. The central notion of such an algorithm is that of a type, which is defined similarly to the types introduced in Section 5.1.1.

Let $\mathcal{T}$ be a general TBox. It can be seen that $\mathcal{T}$ is equivalent to the TBox

$$\top \sqsubseteq \bigsqcap_{C \sqsubseteq D \in \mathcal{T}} \neg C \sqcup D :$$

see Point (v) of Lemma 2.16 for a similar observation. We can thus assume without loss of generality that general TBoxes $\mathcal{T}$ have the form $\{\top \sqsubseteq C_{\mathcal{T}}\}$. Moreover, we can assume that $C_{\mathcal{T}}$ is in negation normal form (NNF); compare Chapter 4. As in Section 3.4, we use $\mathsf{sub}(C)$ to denote the set of subconcepts of the concept C. If $\mathcal{T}$ is a TBox, we set $\mathsf{sub}(\mathcal{T}) = \mathsf{sub}(C_{\mathcal{T}})$.

Definition 5.8. Let $\mathcal{T}$ be a general TBox. A *type* for $\mathcal{T}$ is a set $\tau \subseteq \mathsf{sub}(\mathcal{T})$ such that the following conditions are satisfied: