

1.4.4 Excursus: PSPACE-completeness and quantified boolean formulas

The complexity classes $PSPACE$, P and NP . The complexity class $PSPACE$ denotes the class of all decision problems that are solvable by a deterministic algorithm where the memory requirements are polynomially bounded. The precise definition of $PSPACE$ is the class of all languages $L \subseteq \Sigma^*$ over some finite alphabet Σ such that there is a deterministic Turing machine \mathcal{T} with input alphabet Σ satisfying the following two conditions:

- (1) \mathcal{T} solves the decision problem for L , i.e., \mathcal{T} accepts any input word $w \in L$ and rejects any word $w \in \Sigma^* \setminus L$, and
- (2) \mathcal{T} is polynomially space-bounded, i.e., there is a polynomial $\wp : \mathbb{N} \rightarrow \mathbb{N}$ such that the number of tape cells that \mathcal{T} visits for an input word of length n is at most $\wp(n)$.

In (1), we can think of accepting computations as those that halt with the answer “yes”, while the rejecting computations terminate with the answer “no”. Condition (2) is equivalent to the requirement that there exist a natural number m and a positive constant C such that \mathcal{T} visits at most Cn^m tape cells for input words of length n . One can safely assume that \mathcal{T} reads the complete input word, in which case $\wp(n) \geq n$ for all $n \in \mathbb{N}$.

The complexity class $NPSPACE$ is defined in the same way as $PSPACE$, with the only difference that \mathcal{T} might be a nondeterministic Turing machine. In this case, \mathcal{T} is said to solve the decision problem for L if for each input word $w \in L$, there exists an accepting computation of \mathcal{T} for w , while all computations of \mathcal{T} for words $w \in \Sigma^* \setminus L$ are rejecting. A nondeterministic Turing machine is called *polynomially space-bounded* if there exists a polynomial \wp such that the number of tape cells that \mathcal{T} visits for input words of length n is at most $\wp(n)$.

$PSPACE$ (as any other deterministic complexity class) is closed under complementation. That is, $L \in PSPACE$ if and only if $\bar{L} = \Sigma^* \setminus L \in PSPACE$. A very prominent and useful result is *Savitch's Theorem*, which states that $PSPACE = NPSPACE$. Thus, to show that a decision problem L belongs to $PSPACE$ it suffices to design a nondeterministic polynomially space-bounded algorithm that solves L (or its complement).

$PSPACE$ -hardness of a decision problem L means that any problem $K \in PSPACE$ is *polynomially reducible* to L . That is, there is a deterministic algorithm that takes an input k for K and constructs an input ℓ for L such that the correct answer for the input k for K is “yes” if and only if the correct answer for the input ℓ for L is “yes” and the time complexity for the construction of ℓ from k is polynomially bounded. $PSPACE$ -completeness means membership in $PSPACE$ and $PSPACE$ -hardness.

The time complexity classes $P = PTIME$ and $NP = NPTIME$ are defined in an analogous way. P denotes the class of all decision problems (formally represented by a language $L \subseteq \Sigma^*$) that are solvable by a deterministic polynomially time-bounded algorithm. Formally this means that exists a deterministic Turing machine \mathcal{T} and a polynomial \wp such that, for each input $w \in \Sigma^*$, the computation of \mathcal{T} for w is accepting if $w \in L$ and rejecting if $w \notin L$ and its length is at most $\wp(|w|)$. The class NP is the nondeterministic analogue to P and collects all decision problems L that can be solved by a nondeterministic algorithm such that

- (1) for each input $w \in L$, at least one computation of \mathcal{T} for w is accepting, while for $w \notin L$ all computations are rejecting, and

- (2) for some polynomial \wp , the length of each computation for some input w of length n is bounded by $\wp(n)$.

NP-hardness of a problem L means that all problems in *NP* are polynomially reducible to L . *NP*-completeness of L means that $L \in NP$ and L is *NP*-hard. Whenever a decision problem L is *NP*-hard and K is a decision problem such that L is polynomially reducible to K then (by the transitivity of polynomial reductions), any problem in *NP* is polynomially reducible to K ; and thus, K is *NP*-hard too.

The time complexity class *coNP* consists of all decision problems L such that the complement problem \bar{L} belongs to *NP*. The complement problem \bar{L} of L has the same inputs as L and requires the answer “no” for some input w if and only if L requires the answer “yes” for w . L is called *coNP*-hard iff \bar{L} is *NP*-hard, and L is called *coNP*-complete iff \bar{L} is *NP*-complete.

Since *PSPACE* subsumes *NP* (with Savitch’s Theorem, we may simply argument with the inclusion $NP \subseteq NPSpace$ which is derived from the trivial fact that within $\wp(n)$ steps at most $\wp(n)$ tape cells can be visited), any *PSPACE*-complete problem is *NP*-hard. Thus, assuming that $PSPACE \neq NP$, the *PSPACE*-complete problems are even harder than *NP*-complete ones.

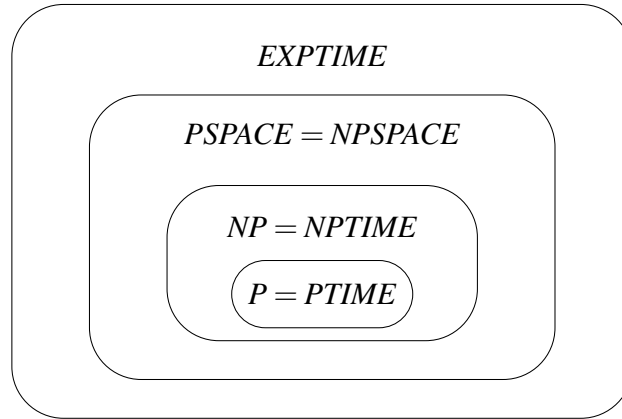


Figure 12: Complexity classes

As in the case of *NP*-completeness, the classical approach to establish *PSPACE*-completeness for L relies on two steps: (1) to show membership to *PSPACE* by providing a polynomially space-bounded (deterministic or non-deterministic) algorithm that solves L and (2) providing a polynomial reduction from some other problem K that is known to be *PSPACE*-hard to L .

Quantified boolean formulas (QBF). One of the most prominent *PSPACE*-complete problems is the problem of checking truth of quantified boolean formulas. Quantified boolean formulas extend propositional formulas by quantifiers \exists and \forall over boolean variables, i.e., variables that stand for one of the truth values 0 (false) or 1 (true). Since the boolean variables serve as atomic formulas, they are often called *atomic propositions*, or briefly *atoms*. In the sequel, we fix a recursively enumerable, nonempty set *Prop* of atomic propositions. That is, the abstract syntax of *quantified boolean formulas* (*QBF* for short) is given by:

$$\varphi ::= \text{true} \mid q \mid \varphi_1 \wedge \varphi_2 \mid \neg \varphi \mid \forall q. \varphi$$

where $q \in Prop$. The constant *false*, other boolean connectives, such as \vee or \rightarrow , and existential quantification are derived as usual, e.g., $false \stackrel{\text{def}}{=} \neg true$ and $\exists q.\varphi \stackrel{\text{def}}{=} \neg \forall q.\neg \varphi$. Given a variable assignment $\mathcal{J} : Prop \rightarrow \{0, 1\}$, $q \mapsto q^{\mathcal{J}} \in \{0, 1\}$ (which assigns truth values to the atomic proposition and plays the role of an *interpretation* for FOL), the satisfaction relation \models is defined in the expected way, i.e.,

$$\begin{aligned} \mathcal{J} &\models true \\ \mathcal{J} &\models q && \text{iff } q^{\mathcal{J}} = 1 \\ \mathcal{J} &\models \varphi_1 \wedge \varphi_2 && \text{iff } \mathcal{J} \models \varphi_1 \text{ and } \mathcal{J} \models \varphi_2 \\ \mathcal{J} &\models \neg \varphi && \text{iff } \mathcal{J} \not\models \varphi \\ \mathcal{J} &\models \forall q.\varphi && \text{iff } \mathcal{J}[q := 0] \models \varphi \text{ and } \mathcal{J}[q := 1] \models \varphi \end{aligned}$$

where $\mathcal{J}[q := 0]$ denotes the variable assignment that agrees with \mathcal{J} , possibly except for q which is evaluated by 0 under $\mathcal{J}[q := 0]$. Then, also the derived operators have the expected meaning. E.g., for existential quantification we get:

$$\mathcal{J} \models \exists q.\varphi \quad \text{iff} \quad \mathcal{J}[q := 0] \models \varphi \text{ or } \mathcal{J}[q := 1] \models \varphi$$

Formula φ is called *satisfiable* if $\mathcal{J} \models \varphi$ for some variable assignment \mathcal{J} . φ is called *valid* if $\mathcal{J} \models \varphi$ for all variable assignments \mathcal{J} . *Equivalence* of two QBF φ_1, φ_2 , denoted $\varphi_1 \equiv \varphi_2$, means that $\mathcal{J} \models \varphi_1$ if and only if $\mathcal{J} \models \varphi_2$ where \mathcal{J} ranges over all variable assignments.

Remark 1.4.16 (QBF vs propositional logic). It is worth noting that any QBF-formula has an equivalent propositional formula. This is due to the observation that

$$\forall q.\varphi \equiv \varphi[q/true] \wedge \varphi[q/false],$$

where $\varphi[q/\psi]$ denotes the formula that results from the syntactic replacement of all free occurrences of q in φ with ψ . Similarly, $\exists q.\varphi$ is equivalent to $\varphi[q/true] \vee \varphi[q/false]$. Thus, QBF is as expressive as propositional logic. However, there are several properties that can easily be expressed in QBF, while any propositional logical formula for them is rather long. More precisely, there exists a sequence $(\varphi_n)_{n \geq 0}$ of quantified boolean formulas such that the φ_n 's are of polynomial length, while equivalent propositional formulas have at least exponential length. ■

Just a few more comments on QBF. Free and bounded occurrences of atomic propositions in QBF are defined in the standard way. That is, $\forall q.\varphi$ binds all occurrences of q in φ . An occurrence of q is called *free* in φ if it is not in the scope of a quantifier $\forall q.(...)$. We write $Free(\varphi)$ for the set of variables that have free occurrences in φ and briefly refer to them as free variables of φ . Clearly, whether $\mathcal{J} \models \varphi$ or $\mathcal{J} \not\models \varphi$ only depends on the assignment for the free atomic propositions, i.e., if \mathcal{J} and \mathcal{J}' are variable assignments that agree on $Free(\varphi)$, then $\mathcal{J} \models \varphi$ iff $\mathcal{J}' \models \varphi$. In particular, if we regard a *QBF-sentence*, i.e., a QBF-formula φ without free variables, then any variable assignment yields the same truth value. In fact, for QBF-sentences the notions of satisfiability and validity have the same meaning. Hence, we may simply speak about the *truth* of a variable-free QBF-formula, and write $\models \varphi$ or $\not\models \varphi$, depending on whether φ holds for all variable assignments or none. For example,

$$\models \forall p \exists q. (p \oplus q), \text{ while } \not\models \exists p \forall q. (p \oplus q).$$

Quantified boolean formulas in *prenex form* have the form $Q_1 q_1 \dots Q_n q_n. \varphi$ where $Q_i \in \{\exists, \forall\}$ and φ is quantifier-free with $Free(\varphi) \subseteq \{q_1, \dots, q_n\}$. Furthermore, we may assume that the q_i 's are pairwise distinct. Using the same techniques as for FOL, any quantified boolean formula φ can be transformed in time $\mathcal{O}(|\varphi|)$ into an equivalent formula in prenex form.

QBF-TRUTH denotes the problem which asks whether a given QBF-sentence is true. Given that there is a polynomial transformation from a given QBF-sentence into an equivalent QBF-sentence in prenex form, one can also suppose that the input of *QBF-TRUTH* are QBF-sentences in prenex form. Although *QBF-TRUTH* seems to be a simpler variant of the satisfiability or validity problem for quantified boolean formulas (that might have free atoms), *QBF-TRUTH* can be viewed as the core problem of several decision problems for quantified boolean formulas. In fact, a quantified boolean formula φ with $Free(\varphi) = \{q_1, \dots, q_n\}$ is

- satisfiable if and only if the QBF-sentence $\exists q_1 \dots \exists q_n. \varphi$ is true,
- valid if and only if the QBF-sentence $\forall q_1 \dots \forall q_n. \varphi$ is true.

Thus, the problems of checking satisfiability or validity for QBF-formulas is polynomially reducible to *QBF-TRUTH*. Then, in fact, also the consequence or equivalence problem for QBF can be solved with a *QBF-TRUTH*-solver.

We now state the prominent result by Stockmeyer and Meyer which can be seen as the QBF-analogue to Cook's Theorem stating the *NP*-completeness of the satisfiability problem of propositional logic:

Theorem 1.4.17 (Theorem by Stockmeyer and Meyer). *QBF-TRUTH is PSPACE-complete.*

Thus, although QBF is as powerful as propositional logic, it can provide more compact (i.e., shorter) representations of boolean condition than propositional logic. The price one has to pay for such compact representations is that decision problems like satisfiability, validity, the consequence or equivalence problem are computationally harder than for propositional logic.

Proof of Theorem 1.4.17. We now turn to the proof of Theorem 1.4.17 (page 62). This amounts showing that *QBF-TRUTH* is solvable in polynomial space (see Lemma 1.4.18) and *QBF-TRUTH* is *PSPACE*-hard (see Lemma 1.4.19). We first address membership to *PSPACE* and sketch an algorithm that checks the truth of a QBF-sentence in prenex form by means of a polynomially space-bounded algorithm. (Recall that any QBF-formula φ can be transformed in time $\mathcal{O}(|\varphi|)$ to an equivalent QBF-formula in prenex form.) In the sequel, let

$$\sigma = Q_1 q_1 \dots Q_n q_n. \varphi$$

where φ is a propositional formula built by the atoms q_1, \dots, q_n and $Q_i \in \{\exists, \forall\}$ for $1 \leq i \leq n$. The truth value of σ can be derived by analysing the *decision tree* of q_1, \dots, q_n . This means a full binary tree of height n where the nodes at level i stand for the assignments for the first i atoms q_1, \dots, q_i and their edges represent the two possible cases $q_{i+1} = 0$ or $q_{i+1} = 1$. Thus, there is a one-to-one correspondence between the leaves of the decision tree and the variable assignments for q_1, \dots, q_n . Hence, in each leaf we can compute the truth value of the matrix φ

of σ according to the variable assignment represented by this leaf. For the QBF-sentence φ to be true, we do not need that all leafs of the full decision tree evaluate to true. Instead, a subtree suffices that results by choosing sons for all nodes at the levels corresponding to an existential quantification in φ . I.e., if $Q_i = \exists$ then for all nodes v at level i , we seek for a truth value $b \in \{0, 1\}$ for q_i such that the formula $Q_{i+1}q_{i+1} \dots Q_n q_n \cdot \varphi$ evaluates to true, when q_1, \dots, q_{i-1} are evaluated according to the variable assignment represented by node v and q_i is evaluated by b . For $Q_i = \forall$, however, we need both branches, i.e., both possibilities $q_i = 0$ and $q_i = 1$ must yield the truth of $Q_{i+1}q_{i+1} \dots Q_n q_n \cdot \varphi$.

This leads to an algorithm which is fairly similar to the decision algorithm for the FO-theory of the ordered rational (see Algorithm 1.4.3 on page 57), and generates a fragment of the decision tree for q_1, \dots, q_n by means of a preorder traversal. Algorithm 1.4.4 on page 64 sketches the main steps of a recursive procedure which uses boolean variables b_1, \dots, b_n to represent the assignments for q_1, \dots, q_n . The initial call is *QBF-check*(0). In fact, the recursion depth is n . Thus, the space requirements of this algorithm are bounded by $\mathcal{O}(|\sigma|) = \mathcal{O}(n + |\varphi|)$. Hence, we obtain:

Lemma 1.4.18. *QBF-TRUTH is solvable by a (deterministic) polynomial space-bounded algorithm.*

The proof of the next lemma shows that each problem $L \in PSPACE$ is polynomially reducible to *QBF-TRUTH*. Thus:

Lemma 1.4.19. *QBF-TRUTH is PSPACE-hard.*

Proof. Let $L \in PSPACE$ and let \mathcal{T} be a deterministic Turing machine (DTM) that decides L in polynomial space. That is, (1) the computation of \mathcal{T} ends in a final state (i.e., \mathcal{T} accepts w) if and only if $w \in L$ and (2) there is a polynomial \wp such that \mathcal{T} visits for all input words w at most $\wp(|w|)$ tape cells. It is no restriction to suppose that $\wp(n) = Cn^r$ for some constants $C \in \mathbb{N}$ and $r \in \mathbb{N}$.

Given an input word w for \mathcal{T} , we will construct a QBF-sentence φ_w such that

- $|\varphi_w| = \mathcal{O}(\text{poly}(|w|))$
- $w \in L$ if and only if φ_w is true.

Since with L also \mathcal{T} is fixed, the size of \mathcal{T} is viewed to be constant.

We suppose here that the tape of \mathcal{T} is bounded to the left which permits to index the tape cells by $0, 1, 2, \dots$. Let Q be the state space of \mathcal{T} and Γ the tape alphabet which includes the blank symbol \sqcup . The starting state of \mathcal{T} will be denoted by q_0 . The set of final states is denoted by Q_F . The transition function of \mathcal{T} is given by a function

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{-1, 0, +1\}$$

The states $q \in Q_F$ are supposed to be terminal, i.e., the computation ends as soon as a state in Q_F will be reached. For technical reasons we suppose that $\delta(q, a) = (q, a, 0)$ for $q \in Q_F$ and all tape symbols $a \in \Gamma$. An input word $w \in (\Gamma \setminus \{\sqcup\})^*$ is accepted by \mathcal{T} iff \mathcal{T} 's computation for w terminates in a state $q \in Q_F$.

Algorithmus 2 Recursive algorithm *QBF-check*(i)

(* Uses global boolean variables ζ_1, \dots, ζ_n and checks whether *)

(* $[q_1 := \zeta_1, \dots, q_i := \zeta_i] \models Q_{i+1} q_{i+1} \dots Q_n q_n \cdot \varphi$. *)

IF $i = n$ **THEN**

Check whether $[q_1 := \zeta_1, \dots, q_n := \zeta_n] \models \varphi$.

If so then return *true*. Otherwise return *false*.

ELSE

IF $Q_{i+1} = \exists$ **THEN**

$\zeta_{i+1} := 0$;

IF *QBF-check*(i + 1) **THEN**

return *true*

ELSE

$\zeta_{i+1} := 1$;

IF *QBF-check*(i + 1) **THEN**

return *true*

FI

FI

return *false*

FI

IF $Q_{i+1} = \forall$ **THEN**

$\zeta_{i+1} := 0$;

IF \neg *QBF-check*(i + 1) **THEN**

return *false*

ELSE

$\zeta_{i+1} := 1$;

IF \neg *QBF-check*(i + 1) **THEN**

return *false*

FI

FI


return *true*

FI

FI

As \mathcal{T} is polynomially space-bounded there exists a polynomial \wp such that the computation of \mathcal{T} for the input word $w \in (\Gamma \setminus \{\sqcup\})^*$ visits at most the tape cells $0, 1, \dots, \wp(n) - 1$ where $n = |w|$ is the length of w . W.l.o.g., we may suppose that $\wp(n) \geq n$.

b	c	a	...	a	c	\sqcup	\sqcup	...
0	1	2						$\wp(n)$


 at most these tape cells
 will be visited

Thus, the configurations that appear in \mathcal{T} 's computation for input w can be described by words

$$a_1 \dots a_l q b_1 \dots b_r \in \Gamma^* Q \Gamma^*, \text{ where } a_i, b_j \in \Gamma, q \in Q \text{ and } l + r = \wp(n).$$

As \mathcal{T} is deterministic, there is exactly one computation for each input word. Furthermore, \mathcal{T} decides L and thus halts for all input words. In particular, the computation of \mathcal{T} for w is finite and visits no configuration twice (as otherwise \mathcal{T} 's computation would be cyclic and therefore infinite). This yields that the length of \mathcal{T} 's computation for w is bounded above by

$$|\Gamma|^{\wp(n)} \cdot |Q| \cdot \wp(n) \quad \text{where } n = |w|.$$

The first factor $|\Gamma|^{\wp(n)}$ stands for the potential tape contents, the factor $|Q|$ for the possible states and the factor $\wp(n)$ for the possible positions of the cursor.

The construction of the QBF-sentence φ_w has some similarities with Cook's proof for the NP -hardness of SAT (the satisfiability problem of propositional logic). However, we have here a space-bound, rather than a time-limit. The QBF-sentence φ_w has the following form:

$$\varphi_w \stackrel{\text{def}}{=} \exists \bar{p} \exists \bar{p}'. \left(\text{init}(\bar{p}) \wedge \text{reach}(\bar{p}, \bar{p}') \wedge \text{accept}(\bar{p}') \right)$$

where \bar{p} and \bar{p}' are tuples of atomic propositions that serve to encode the configurations of \mathcal{T} . As for FOL, we use the notation $\varphi(\bar{q})$ to denote that the free variables of φ are contained in the tuple \bar{p} . The intuitive meaning of the subformulas of φ_w is as follows:

- $\text{init}(\bar{p})$ states that \bar{p} encodes the initial configuration $q_0 w \sqcup^{\wp(n)-|w|}$,
- $\text{accept}(\bar{p}')$ asserts that \bar{p}' encodes a final configuration of the form $v_l q v_r$ where $q \in Q_F$.
- $\text{reach}(\bar{p}, \bar{p}')$ states that the configuration encoded by \bar{p}' is reachable from the configuration encoded by \bar{p} by \mathcal{T} 's computation.

To define these three QBF-formulas, we use three types atomic propositions:

- atoms $\text{tape}(j, a)$ for $0 \leq j < \wp(n)$ and $a \in \Gamma$ stating that in the current configuration tape cell j contains the symbol a ,
- atoms $\text{state}(q)$ for $q \in Q$ stating that the current state is q
- atoms $\text{pos}(j)$ for $0 \leq j < \wp(n)$ stating that the cursor currently points to tape cell j .

The tuple consisting of these atoms is denoted by \bar{p} . Note that the number of atoms in \bar{p} is

$$M \stackrel{\text{def}}{=} |\Gamma| \cdot \wp(n) + |Q| + \wp(n),$$

and therefore polynomial in the length of w .

The tuple \bar{p}' is also an M -tuple consisting of copies of these atoms. These copies will be denoted in primed notation, say $tape'(j, a)$, $state'(q)$ and $pos'(j)$. Furthermore, we will use some more copies of \bar{p} . These will be denoted by \bar{r} , \bar{q} and \bar{q}' . We will use some simplifying notations as for FOL: e.g., given QBF-formula $\varphi(\bar{p})$ and a variable tuple \bar{q} that has the same length as \bar{p} then we simply write $\varphi(\bar{q})$ to denote the formula $\varphi[\bar{p}/\bar{q}]$ that results from φ by replacing all free occurrence of the i -th variable of \bar{p} with the i -th variable of \bar{q} .

The formula for the starting configuration is as follows. Let $w = b_0 b_1 \dots b_{n-1}$ be the input word where the b_i 's are letters in $\Gamma \setminus \{\sqcup\}$. Then, the initial configuration is

$$q_0 b_0 b_1 \dots b_{n-1} \sqcup^{\wp(n)-n}.$$

This is encoded by the QBF-formula:

$$init(\bar{p}) \stackrel{\text{def}}{=} state(q_0) \wedge pos(0) \wedge \bigwedge_{0 \leq j < n} tape(j, b_j) \wedge \bigwedge_{n \leq j < \wp(n)} tape(j, \sqcup)$$

The length of $init(\bar{p})$ is

$$\mathcal{O}(\wp(n) \cdot |\Gamma| + |Q|),$$

and therefore polynomial in $n = |w|$. Recall that \wp is a polynomial and that $|\Gamma|$ and $|Q|$ are treated as constants (since \mathcal{T} is fixed).

The acceptance condition is formalized by the QBF-formula:

$$accept(\bar{p}') \stackrel{\text{def}}{=} \bigvee_{q \in Q_F} state'(q)$$

which is of constant length as \mathcal{T} is supposed to be fixed. It remains to provide the definition of the formula $reach(\bar{p}, \bar{p}')$. We will use a recursive definition with several copies of \bar{p} and \bar{p}' . The idea for the definition of $reach(\bar{p}, \bar{p}')$ relies on QBF-formulas $\varphi_i(\bar{p}, \bar{p}')$ stating that \mathcal{T} in the configuration encoded by \bar{p} will reach the configuration encoded by \bar{p}' in 2^i or fewer steps. Recall that the number of steps in the computation of \mathcal{T} for the input word w is at most

$$K \stackrel{\text{def}}{=} |\Gamma|^{\wp(n)} \cdot |Q| \cdot \wp(n)$$

where $n = |w|$ (see above). We then put $k \stackrel{\text{def}}{=} \lceil \log K \rceil$ and define

$$reach(\bar{p}, \bar{p}') \stackrel{\text{def}}{=} \varphi_k(\bar{p}, \bar{p}').$$

Note that k is polynomially bounded in $n = |w|$ as we have:

$$\begin{aligned} k &= \lceil \log K \rceil \\ &= \lceil \log(|\Gamma|^{\wp(n)} \cdot |Q| \cdot \wp(n)) \rceil \\ &\leq \wp(n) \log |\Gamma| + \log |Q| + \log \wp(n) + 1 \\ &= \mathcal{O}(poly(n)) \end{aligned}$$

The definition of $\varphi_i(\bar{p}, \bar{p}')$ will use the following subformulas:

- $\text{conf}(\bar{p})$ is a QBF-formula that asserts that \bar{p} encodes a “legal” configuration of \mathcal{T} by stating that
 - exactly one of the atoms $\text{state}(q)$ is fulfilled,
 - for each position j exactly one of the atoms $\text{tape}(j, b)$ holds,
 - and there is exactly one position j where $\text{pos}(j)$ holds.

The precise definition of $\text{conf}(\bar{p})$ is left as an exercise.

- For $q \in Q$ and $a \in \Gamma$, let $\varphi_{\delta, q, a}(\bar{p}, \bar{p}')$ be a QBF-formula that formalizes the effect of the transition function δ when the current state is q and the symbol under the cursor is a . More precisely, QBF-formula $\varphi_{\delta, q, a}(\bar{p}, \bar{p}')$ asserts that if \bar{p} encodes a configuration of the form $v q a v'$ then \bar{p}' is the encoding for the (unique) successor configuration of $v q a v'$. Suppose, for instance, that $\delta(q, a) = (p, b, +1)$. Then:

$$v q a v' \vdash_{\mathcal{T}} v b p v'$$

which is formalized by the following QBF-formula:

$$\varphi_{\delta, q, a}(\bar{p}, \bar{p}') \stackrel{\text{def}}{=} \bigwedge_{0 \leq j < \wp(n)} \left(\text{state}(q) \wedge \text{pos}(j) \wedge \text{tape}(j, a) \rightarrow \text{state}'(q) \wedge \text{pos}'(j+1) \wedge \text{tape}'(j, b) \wedge \text{unchanged}(j) \right)$$

where $\text{unchanged}(j)$ states that the content of the cells $i \neq j$ are unchanged, i.e.,

$$\text{unchanged}(j) \stackrel{\text{def}}{=} \bigwedge_{\substack{0 \leq i < \wp(n) \\ i \neq j}} \bigwedge_{a \in \Gamma} (\text{tape}(i, a) \leftrightarrow \text{tape}'(i, a))$$

The definition of $\varphi_{\delta, q, a}$ is similar when $\delta(q, a) = (p, b, 0)$ or $\delta(q, a) = (p, b, -1)$.

As we assume that $\delta(q, a) = (q, a, 0)$ if $q \in Q_F$ we have

$$\varphi_{\delta, q, a}(\bar{p}, \bar{p}') \models \bar{p} = \bar{p}'$$

where $\bar{p} = \bar{p}'$ abbreviates the formula $\bigwedge_i (p_i \leftrightarrow p'_i)$ where p_i and p'_i are the i -th components of \bar{p} and \bar{p}' , respectively.

- QBF-formula $\text{step}(\bar{p}, \bar{p}')$ states that \mathcal{T} moves from the configuration encoded by \bar{p} to configuration encoded by \bar{p}' . Furthermore, we integrate the condition that \bar{p} and \bar{p}' are indeed configurations. For instance, we may deal with the following QBF-formula:

$$\text{step}(\bar{p}, \bar{p}') \stackrel{\text{def}}{=} \text{conf}(\bar{p}) \wedge \text{conf}(\bar{p}') \wedge \bigwedge_{q \in Q} \bigwedge_{a \in \Gamma} \varphi_{\delta, q, a}(\bar{p}, \bar{p}')$$

Note that the length of formula $\text{step}(\bar{p}, \bar{p}')$ is polynomially bounded in $n = |\omega|$.³

³With the presented definition of $\text{step}(\bar{p}, \bar{p}')$ that contains both subformulas $\text{conf}(\bar{p})$ and $\text{conf}(\bar{p}')$, the resulting formulas $\varphi_i(\bar{p}, \bar{p}')$ contain redundant constraints for the variable-tuples to be proper encodings of configurations. The redundancies could be avoided, but are irrelevant for the asymptotic length of the constructed QBF-formulas.

A first attempt to define $\varphi_i(\bar{p}, \bar{p}')$ could now be to define

$$\varphi_0(\bar{p}, \bar{p}') \stackrel{\text{def}}{=} (\bar{p} = \bar{p}') \vee \text{step}(\bar{q}, \bar{q}')$$

and, recursively,

$$\varphi_{i+1}(\bar{p}, \bar{p}') \stackrel{\text{def}}{=} \exists \bar{r}. \left(\varphi_i(\bar{p}, \bar{r}) \wedge \varphi_i(\bar{r}, \bar{p}') \right) \quad (*)$$

Indeed, $\varphi_i(\bar{p}, \bar{p}')$ holds exactly for those valuations for \bar{p} and \bar{p}' such that \mathcal{T} moves within at most 2^i steps from \bar{p} to \bar{p}' . This yields that the QBF-sentence φ_w is true if and only if the computation of \mathcal{T} for input w ends in a final state. As \mathcal{T} decides the *PSPACE*-language L we get the desired connection:

$$\varphi_w \text{ is true} \quad \text{iff} \quad w \in L$$

However, the transformation $w \mapsto \varphi_w$ is *not* polynomial since the length of the φ_i 's grows *exponentially* in i . Thus, the length of $\text{reach}(\bar{p}, \bar{p}')$ with the above definition is exponential in the length of w . (Suppose for simplicity that $|\Gamma| = 2$. Then, $K \geq |\Gamma|^{\wp(n)}$ and $k = \lceil \log K \rceil \geq \wp(n)$. But then $|\varphi_k| \geq 2^k \geq 2^{\wp(n)}$.)

We therefore redefine the formulas φ_i in a more space-efficient way. The formula $\varphi_0(\bar{p}, \bar{p}')$ is as before. For $i \geq 0$, we redefine:

$$\varphi_{i+1}(\bar{p}, \bar{p}') \stackrel{\text{def}}{=} \exists \bar{r} \forall \bar{s} \forall \bar{s}'. \left(\left((\bar{s}, \bar{s}') = (\bar{p}, \bar{r}) \vee (\bar{s}, \bar{s}') = (\bar{r}, \bar{p}') \right) \rightarrow \varphi_i(\bar{s}, \bar{s}') \right) \quad (**)$$

The trick with the above definition is to introduce new tuples of atoms \bar{s} and \bar{s}' that allow to “reuse” the same subformula $\varphi_i(\bar{s}, \bar{s}')$ to express both conditions $\varphi_i(\bar{p}, \bar{r})$ and $\varphi_i(\bar{r}, \bar{p}')$. With this trick we avoid a duplication of the formula-length. In fact, the formulas φ_i defined by (**) are equivalent to the formulas φ_i defined by (*). Thus, they also yield the desired result stating that φ_w is true if and only if $w \in L$. Furthermore, the length of φ_i is given by the recurrence

$$|\varphi_{i+1}| = \mathcal{O}(M) + |\varphi_i|$$

where M denotes the number of atoms in the tuples \bar{p} . The solution of the above recurrence is $|\varphi_i| = \mathcal{O}(M \cdot i)$. As mentioned before, we have:

$$M = (|\Gamma| + 1) \cdot \wp(n) + |Q| = \mathcal{O}(\text{poly}(n))$$

This yields that the length of φ_k , and therefore also the length of φ_w , is polynomially bounded in $n = |w|$. Hence, the transformation $w \mapsto \varphi_w$ provides a polynomial reduction from the word-problem for L to *QBF-TRUTH*. \square

Lemma 1.4.18 and 1.4.19 complete the proof of Theorem 1.4.17 (page 62) stating the *PSPACE*-completeness of *QBF-TRUTH*.

1.4.5 PSPACE-completeness of the FO-theory of the ordered rationals

In Section 1.4.3 on page 53 ff, we considered the FO-theory $Th(\mathbb{Q}, \leq)$, i.e., the theory consisting of all FOL-formulas that hold for the ordered rationals. The underlying vocabulary consists of a single binary predicate symbol \leq that stands for the natural “less-or-equal” relation on the rationals. In Section 1.4.3 we saw that $Th(\mathbb{Q}, \leq)$ is decidable, but we did not discuss the complexity. This section is devoted to provide a proof for the following theorem:

Theorem 1.4.20. *The FO-theory of (\mathbb{Q}, \leq) is PSPACE-complete.*

Since the recursive algorithm sketched in Algorithm 1.4.3 on page 57 for checking whether a given FOL-sentence (over the vocabulary consisting of a binary predicate symbol \leq) holds in (\mathbb{Q}, \leq) is polynomially space-bounded, we get:

Lemma 1.4.21. *The FO-theory of (\mathbb{Q}, \leq) belongs to PSPACE.*

It remains to show PSPACE-hardness of $Th(\mathbb{Q}, \leq)$. For this, we apply the classical *reduction principle* for PSPACE-hardness proofs and provide a polynomial reduction from QBF-TRUTH.

Lemma 1.4.22. *The FO-theory of (\mathbb{Q}, \leq) is PSPACE-hard.*

Proof. We provide a polynomial reduction from QBF-TRUTH to the question whether a given FOL-sentence over the vocabulary with a single binary predicate \leq holds in (\mathbb{Q}, \leq) . So, we start with a QBF-sentence in prenex form

$$\sigma = Q_1 p_1 \dots Q_n p_n \cdot \varphi$$

where φ is a boolean formula (i.e., formula of propositional logic) with the boolean variables $p_1, \dots, p_n \in Prop$ and $Q_i \in \{\forall, \exists\}$ for $1 \leq i \leq n$. We aim to construct in time $\mathcal{O}(\text{poly}(|\sigma|)) = \mathcal{O}(\text{poly}(n + |\varphi|))$ a FOL-sentence ϕ such that

$$(\mathbb{Q}, \leq) \models \phi \text{ if and only if } \sigma \text{ is true.}$$

The idea of this transformation $\sigma \mapsto \phi$ is to replace the atomic propositions p_i with atomic FOL-formulas $x_i \leq 0$ and to replace the p_i ’s in the quantifier prefix with the x_i ’s. For example, the quantified boolean tautology

$$\sigma = \forall p_1 \exists p_2. (p_1 \oplus p_2)$$

is transformed into the FOL-sentence

$$\phi = \forall x_1 \exists x_2. ((x_1 \leq 0) \oplus (x_2 \leq 0)),$$

which, in fact, is true in (\mathbb{Q}, \leq) since for each rational number r_1 as value for x_1 , we may pick a rational number r_2 as value for x_2 such that $(r_1 \leq 0) \oplus (r_2 \leq 0)$ evaluates to *true*. (Just pick $r_2 = -r_1$ if r_1 is non-zero and $r_2 = 1$ if $r_1 = 0$.)

Instead of the constant 0, we may deal with any fixed rational number. However, since our vocabulary does not have constant symbols we simply deal with an existential quantified variable y rather than 0. So, the definition of ϕ is as follows. We pick pairwise distinct first-order variables x_1, \dots, x_n, y and consider the FOL-formula

$$\phi \stackrel{\text{def}}{=} \exists y Q_1 x_1 \dots Q_n x_n. \varphi[p_1/(x_1 \leq y), \dots, p_n/(x_n \leq y)]$$

where $\varphi[p_1/(x_1 \leq y), \dots, p_n/(x_n \leq y)]$ denotes the formula that results from φ by the uniform syntactic replacement of all occurrences of the boolean variable p_i in φ with the atomic FOL-formula $x_i \leq y$. Then,

$$|\phi| = \mathcal{O}(1 + n + |\varphi|) = \mathcal{O}(|\sigma|)$$

and the FOL-sentence ϕ holds in (\mathbb{Q}, \leq) if and only if the QBF-sentence σ is true. \square