社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－AL－120 (8)
2008／11／7

交代性多項式時間における別解問題

八登 崇之
東京大学大学院情報理工学系研究科
コンピュータ科学専攻
yato@is.s.u-tokyo.ac.jp

概 要

別解問題 (ASP) というのは、問題のインスタンスとともに解が与えられた時に他の解を見つけるというものである。ASP の計算量解析に重要な役割を果たすものとして ASP 完全性があり、これは関数問題（多価関数）に対して定義される。本研究では、計算量分野で有名な人工的二人ゲームである QBF ゲームが **PSPACE** に対応するある関数問題クラスにおいて ASP 完全となることを示す。当該のクラスでは一人パズルゲーム「倉庫番」の三次元版が ASP 完全になることが既に示されている。このことは、多項式手数の二人ゲームと超多項式手数の一人ゲームの等価性が言語クラスのみならず関数問題クラスにおいても成立することを示唆している。

Another Solution Problems in Alternating Polynomial Time

Takayuki YATO
Department of Computer Science
Graduate School of Information
Technology and Science
The University of Tokyo
yato@is.s.u-tokyo.ac.jp

Abstract

The Another Solution Problem (ASP) is the problem to find a solution other than the given ones. One of the important tools for analyzing the computational complexity of ASPs is ASP-completeness, which is defined for function problems (multivalued functions). This study shows that QBF Game, an artificial two-player game well known in the complexity, is ASP-complete in a function problem class which corresponds to **PSPACE** and in which (a version of) Sokoban is also ASP-complete. This result suggests that the equivalence between two-player games with polynomial duration and one-player puzzles with superpolynomial duration holds in the world of the function problems.

1 Introduction

For a function problem X, ASP of X denotes the following problem: given an instance of X and some solutions to the instance, find a solution other than the given ones. (Here function problems are formulated as *multivalued* functions.) ASPs of some specific problems have appeared sparsely in the context of computational complexity. For instance, Papadimitriou [2] shows that the ASP of the Hamiltonian Circuit problem is **NP**-complete in contrast to the fact that the restricted case for cubic graphs is trivial.

Ueda and Nagao [7], in contrast, first regarded ASP as a new class of problems (and used the term ASP) in their complexity analysis of a type of puzzle Nonogram. Moreover they pointed out that ASP has close relation to the task of designing problems for puzzles requiring uniqueness of solutions, since the work for checking a newly made puzzle problem does not have any solution other than intended one can clearly be viewed as an instance of ASP. They also showed that a special kind of reduction is a useful tool for examining complexity of ASPs; that is a parsimonious reduction that allows efficient transformation of solutions. (Similar to *function-parsimonious* reductions that we use in this study but in weaker form.)

Later Yato and Seta [8] introduced the notion of *ASP-completeness*, which is the completeness with respect of the 'strongly' parsimonious reducions, and proved the completeness (in class **FNP**) of some well-known puzzle types, including Sudoku. The fact that a function problem X is ASP-complete in **FNP** implies the **NP**-completeness of the decision version of X as well as the **NP**-completeness of the ASP of X. (Note that the original problem can be viewed as a special case of ASP where

the number of given solutions is zero.) The way to prove the ASP-completeness of X is to establish a function-parsimonious reduction from some 'canonical' complete problem such as 3SAT to X. This method has an advantage in that it often enables one to prove the **NP**-completeness of ASPs using existing results of reductions without actually considering the ASPs at all since in many cases reductions between languages can be easily modified to be function-parsimonious.

It is worthy of noticing that function-parsimonious reductions are defined in function problems rather than languages and thus ASP-completeness is defined for function problem classes (such as **FNP**). So far function problem classes have been studied mostly from theoretical interest (refer to Selman's survey [3] for such studies). Our study is different in that it is aimed at the analysis of ASP, which has practical interest in the realm of puzzle design.

However, there are popular and amazing puzzles outside the level of **NP** in our world. One type of such examples is those regarding motions. This type includes block-pushing puzzle such as 'Sokoban' and block-moving puzzle such as 'Ruch Hour'. The decision of solvablity of such puzzles lie in the class **PSPACE** due to the fact that the length of solution can grow superpolynomially. Seta and Yato [4] extended the theory of ASP to be applicable for the problems outside **NP**. One of the major obstacles in such extension is absence of the suitable definition of multivalued function classes that correspond to the higher language classes. They introduced a new type of transducer, called *OB-transducer*, which is a transducer with restriction on the way to output. Then they defined a class $\mathbf{FNPSPACE}_{\text{p-out}}$ using OB-transducers and proved that the three-dimensional version of Sokoban is ASP-complete in that class. The class $\mathbf{FNPSPACE}_{\text{p-out}}$ is so carefully defined that it captures the nature of Sokoban and other block-moving puzzles in that the solution can be extremely long but verifiable in polynomial time in output length. The ASP-completeness signifies the defined class suitably captures the nature of Sokoban and its solution definition.

The other type of puzzle that lies outside **NP** is finding a winning strategy for the player to move next when given a position of a two-player game, such as Chess studies and Tsume-Shogi. Then we use as a natural definition of solutions a tree called trace as defined in [5]. Two-player games are in complexity theory formalized as alternating computation [1]. When a game has duration (the number of moves) bounded in polynomial in input length then it corresponds to alternating polynomial time computation, and the class of languages which can be decided by such computation coincides with **PSPACE** [1], which is the same as one-player puzzle with superpolynomial duration such as Sokoban. Then it will be of interest whether this equality holds in the world of function problems: Our present study answers this question in affirmative way. That is, we prove that QBF-game (an artificial two-player game with polynomial duration) is ASP-complete when a modified form of trace is employed as solutions.

2 Preliminaries

2.1 Function Problems

We follow the framework for function problems and their complexity that is proposed by Selman [3].

Let Σ^* be a fixed alphabet. A function problem is formulated as a *partial multivalued function* from Σ^* to Σ^* (we simply call it a *multivalued function* thereafter).

For a multivalued function f, we write $f(x) \mapsto y$ when y is a value of $f(x)$ (that is, y is a solution to instance x). Moreover, we define the following notations:

- the *solution set* of f to an instance x: $\mathit{set}\text{-}f(x) \stackrel{\text{def}}{=} \{y \mid f(x) \mapsto y\}$.
- the *graph* of f: $\mathit{graph}(f) \stackrel{\text{def}}{=} \{\langle x, y\rangle \mid f(x) \mapsto y\}$.
- the *domain* of f: $\mathit{dom}(f) \stackrel{\text{def}}{=} \{x \mid \exists y.\, f(x) \mapsto y\}$.

A multivalued function f is called *polynomially-balanced* when there is a polynomial p such that $|y| \leq p(|x|)$ holds for all x and y satisfying $f(x) \mapsto y$. A *exponentially-balanced* function is defined in an analogous way.

A computation model for multivalued functions is a *transducer*, which is a Turing machine with a write-only output tape. A transducer may be determinstic or nondeterministic. For a transducer M, we define a multivalued function $M(\cdot)$ as follows: $M(x) \mapsto y$ if and only if M halts in an accepting state on an input x and leaves y on the output tape when it halts. When M is nondeterministic, the function $M(\cdot)$ is multivalued generally. We say that M *represents* a multivalued function f when $M(\cdot)$ is equal to f.

2.2 Two-Player Games

Here we give a formal definition of two-person games, based on [5].

Definition 2.1. A *(two-person perfect information) game* G is a triple (P_1, P_2, R), where

- P_1 (resp. P_2) is the set of positions where Player 1 (resp. 2) is to move.

- R is a subset of $(P_1 \times P_2) \cup (P_2 \times P_1)$, and means the set of all legal moves.

The winning positions of Player 1 is defined as follows.

Definition 2.2. Let $G = (P_1, P_2, R)$ be a game. The set of *winning positions (with respect to Player 1)* of G, denoted as $W(G)$, is defined as follows:

- $W_{-1}(G) := \emptyset$.
- $W_i(G) := W_{i-1}(G)$
  $\cup \left\{ \pi_1 \in P_1 \;\middle|\; \exists \pi_2 \in P_2 \left(\begin{array}{l} (\pi_1 \to \pi_2) \in R \\ \wedge\, \pi_2 \in W_{i-1}(G) \end{array} \right) \right\}$
  $\cup \left\{ \pi_2 \in P_2 \;\middle|\; \forall \pi_1 \in P_1 \left(\begin{array}{l} (\pi_2 \to \pi_1) \in R \\ \Longrightarrow \pi_1 \in W_{i-1}(G) \end{array} \right) \right\}$.
- $W(G) := \bigcup_{i \geq 0} W_i(G)$.

Then a tree T constructed as follows (called *trace* there) is considered as solution of a two-person game G:

- T is a part of AND/OR tree consisting of the positions in $W(G)$ (the winning positions of Player 1).
- A position π of Player 1's turn in T has one child π' such that $\pi \to \pi'$ is legal.
- A position π of Player 2's turn in T has as children all the positions π' such that $\pi \to \pi'$ is legal.

Definition 2.3. The *QBF Game* is the game (P_1, P_2, R), where P_1, P_2, R is defined as follows:

- P_1 is the set of all quantified boolean formulas in prenex form with fully alternating quantifiers starting with $\exists$.
- P_2 is the same as P_1, except that quantifiers start with $\forall$.
- $R = \{(\pi.\pi') \in P_1 \times P_2 \mid \pi \to \pi'\} \cup \{(\pi'.\pi) \in P_2 \times P_1 \mid \pi \to \pi'\}$; where legal moves $\pi \to pi'$ is declared as follows: $\pi \to pi'$ holds if and only if π' is consructed from π by removing its first quantified variable, assigning 0 or 1 to that variable and reducing the body by following rules: $\not 0 = 1$, $0 \wedge x = 0$, $0 \vee x = x$, etc. There is exceotions when the reduced expression results in constants: Player 1 cannot move to constant 0 and Player 2 cannot move to constant 1. Note that if an assigned value of a variable is irrevant in the body expression then there is only one choice.

3 Theory of Another Solution Problem

3.1 Definition of ASP

In this section we describe the theory of Another Solution Problem (ASP).

Definition 3.1. Let f be a multivalued function. The *n-ASP* of f, denoted ASP^n-f, is the multivalued function described as follows:

- A well-formed instance of ASP^n-f is (the string which encodes) a pair (x, S) where $x \in \Sigma^*$ is an instance of f, $|S| = n$ and $S \subseteq$ *set-*$f(x)$.
- The solution set is defined as follows:

$$\textit{set-}(\mathrm{ASP}^n\text{-}f)(x, S) \stackrel{\text{def}}{=} \textit{set-}f(x) - S. \quad (1)$$

One of our main tools for investigating the complexity of ASPs is *function-parsimonious reductions*, which are parsimonious reductions with the condition that the transformation of solutions is tractable.

Definition 3.2. Let f and g be multivalued functions. A *function-parsimonious reduction* from f to g is a triple $(\phi_{\mathrm{I}}, \phi_{\mathrm{F}}, \phi_{\mathrm{B}})$ satisfying the following conditions:

- Let $\phi_{\mathrm{I}}(x) = x'$. Then there is a bijection ϕ_x from *set-*$f(x)$ to *set-*$g(x')$.
- If $f(x) \mapsto y$ then $\phi_{\mathrm{F}}(x, y) = \phi_x(y)$.
- If $f(x') \mapsto y'$ then $\phi_{\mathrm{B}}(x, y') = \phi_x^{-1}(y')$.

When all of ϕ_{I}, ϕ_{F}, and ϕ_{B} are computable in polynomial time, we say $(\phi_{\mathrm{I}}, \phi_{\mathrm{F}}, \phi_{\mathrm{B}})$ is a polynomial-time function parsimonious reduction from f to g and write $f \leq_{\mathrm{f\#}} g$ via $(\phi_{\mathrm{I}}, \phi_{\mathrm{F}}, \phi_{\mathrm{B}})$.

The existence of bijection ϕ_x implies that the instance transformer ϕ_{I} preserves the number of solutions and thus function-parsimonious reducions are a special case of parsimonious reducions. ϕ_{F} and ϕ_{B} are the forward and backward solution transformers respectively.

Proposition 3.1. *Let f and g be multivalued functions. If $f \leq_{\mathrm{f\#}} g$ then ASP^n-$f \leq_{\mathrm{f\#}} \mathrm{ASP}^n$-$g$ for any nonnegative integer n.*

Proposition 3.2. *For any multivalued function f and nonnegative integers m and n, we have ASP^m-ASP^n-$f \leq_{\mathrm{f\#}} \mathrm{ASP}^{m+n}$-$f$*

Combining the two propositions above, we obtain the following important result.

Theorem 3.3. *Let f be a multivalued function. If $f \leq_{\mathrm{f\#}} \mathrm{ASP}^1$-$f$, then for any nonnegative integer n we have $f \leq_{\mathrm{f\#}} \mathrm{ASP}^n$-$f$.*

3.2 ASP-completeness

ASP-completeness is completeness with respect to function-parsimonious reductions, and defined as follows:

Definition 3.3. Let f be a multivalued function and $\mathcal{F}$ be a function class. A multivalued function f is *ASP-complete in $\mathcal{F}$ (ASP-$\mathcal{F}$-complete)* if and only if $f \in \mathcal{F}$ and $g \leq_{\mathrm{f\#}} f$ for any $g \in \mathcal{F}$.

This definition of ASP-completeness is slightly modified from that used in [8] in two points: (1) we use function-parsimonious reductions instead of old 'ASP-reductions'; (2) ASP-completeness is defined for any function class not restricted to **FNP**.

The following propositions are straightforward from the definitions.

Proposition 3.4. *Let f and g be function problems in a class $\mathcal{F}$. If a problem f is ASP-$\mathcal{F}$-complete and $f \leq_{\mathrm{f\#}} g$ holds then g is also ASP-$\mathcal{F}$-complete.*

First we show the most important property of ASP-completeness.

Theorem 3.5. *Let $\mathcal{F}$ be a function class and suppose there is a multivalued function f satisfying the following condition:*

$$f \text{ is ASP-}\mathcal{F}\text{-complete and } f \leq_{\mathrm{f\#}} \mathrm{ASP}^1\text{-}f. \quad (2)$$

Then for any ASP-$\mathcal{F}$-complete multivalued function g and any nonnegative integer n the problem ASP^n-g is ASP-$\mathcal{F}$-complete.

Proof. From $f \leq_{\mathrm{f\#}} \mathrm{ASP}^1$-$f$ and Proposition 3.3 we have $f \leq_{\mathrm{f\#}} \mathrm{ASP}^n$-$f$. Since f and g is ASP-$\mathcal{F}$-complete we have $f \leq_{\mathrm{f\#}} g$ and applying Proposition 3.1 to it we obtain ASP^n-$f \leq_{\mathrm{f\#}} \mathrm{ASP}^n$-$g$. Combining both relations shows $f \leq_{\mathrm{f\#}} \mathrm{ASP}^n$-$g$, and since f is ASP-$\mathcal{F}$-complete g is so. ∎

In many cases some 'canonical' ASP-complete problems in a class are expected to satisfy the property (2) and in such a class ASP-completeness of a multivalued function automatically derives ASP-completeness of n-ASP.

Next we argue the relation of ASP-completeness and completeness of languages.

Proposition 3.6. *Let $\mathcal{F}$ be a function class. When a multivalued function f is ASP-$\mathcal{F}$-complete, $dom(f)$ is $dom(\mathcal{F})$-complete (under many-one reductions). Here $dom(\mathcal{F})$ is defined as $\{dom(f) \mid f \in \mathcal{F}\}$.*

From Theorem 3.5 and Propostion 3.6, we have the following proposition:

Corollary 3.7. *Let $\mathcal{F}$ be a function class and suppose there is a multivalued function f satisfying (2). Then for any ASP-$\mathcal{F}$-complete multivalued function g and any nonnegative integer n $dom(\mathrm{ASP}^n$-$g)$ is $dom(\mathcal{F})$-complete.*

4 Function version of PSPACE

Here we provide a function class that is analogous to **PSPACE** and captures the nature of trace finding.

4.1 OB-transducers

OB-transducers are transducers which has a restriction on the way that they output a symbol and defined as follows:

Definition 4.1. An *OB-transducer* is a nondeterministic transducer M which satisfies the following conditions:

- The states of finite state control are classified into four types: accepting states, rejecting states, deterministic states and nondeterministic states. When M enters either of the former two types of state, M halts.
- A transition from a deterministic state is always deterministic and M does not write any symbol to its output tape in such a transition.
- A transition from a nondeterministic state has nondeterministic options, and there is a one-to-one correspondence between the set of options and the alphabet Σ. Whenever a option is chosen the corresponding symbol in Σ is written to the output tape. That is, the transition function can be formulated as a function that maps a pair of a local configuration and an output symbol to the next local configuration.

The multivalued function f which M represents is defined in the same way as ordinary nondeterministic transducers: $M(x) \mapsto y$ if and only if M halts in an accepting state on an input x with leaving y on the output tape.

Although the definition requires that moves with output must be nondeterminitic, an OB-transducer actually can output a symbol a deterministically, by branching nondeterministically and immediately rejecting on all the branches but the one that output a.

The next fact suggests importance of considering OB-transducers.

Proposition 4.1. *The class of multivalued functions which can be represented by an OB-transducer running in polynomial time is equivalent to* **FNP**.

An important property of **FNP** problems as opposed to **NPMV** ones are that they can be verified in polynomial time. This property relates the problem to solve a Sudoku-like puzzle to the class **FNP**, since solutions of such a puzzle can also be verified easily. When we think of the problem of finding traces, we notice the following property: although the length of a solution can be exponential in the instance size, it can always be verified in polynomial time with respect to the sum of the lengths of the instance and the solution. It means that the class of multivalued functions representable by polynomial-space transducers is not suitable since the graph of a multivalued function in such a class does not always belong to **P** (unless **P** = **PSPACE**).

The reason that a multivalued function in such classes is not verifiable efficiently is that some information on which choice a transducer took in nondeterministic branches does not appear in the resulted output. If the choice is always recorded in the output, then the verification of a solution can be done in the same time as the transducer took for outputting the solution. Hence the Proposition 4.1 holds.

As for multivalued functions corresponding to **PSPACE**, however, we have another issue. Even if the time needed to verify a function can be bounded by the running time of an OB-transducer that represents the function, the running time of polynomial-space bounded OB-transducers is not bounded in polynomial neither in the input length nor the output length: it could be arbitrarily large compared to the input since a machine could loop many times and then halt, and it could be exponentially long in the output length since a machine could continue to run long deterministically without outputting a symbol. Thus we need a further restriction that the running time of the OB-transducer is not extremely long compared to the *output length.*

4.2 Definition of the class $\mathbf{FNPSPACE}_{\text{p-out}}$

From the argument above we define a function problem version of **PSPACE** using OB-transducers as follows:

Definition 4.2.
- We define **FNPSPACE** as the class of multivalued functions f which satisfies the following condition: there exists a nondeterministic transducer M which represents f and runs in exponential time and polynomial space with respect to the input length.
- We define $\mathbf{FNPSPACE}_{\text{p-out}}$ as the class of multivalued functions f which satisfies the following condition: there exists a nondeterministic transducer M which represents f and runs in exponential time and polynomial space with respect to the input length and in polynomial time with respect to the length of an output.

The class **FNPSPACE** is a natural **PSPACE** analogue of **NP**, but here we utilize the class $\mathbf{FNPSPACE}_{\text{p-out}}$ which imposes an additional restiction (hence $\mathbf{FNPSPACE}_{\text{p-out}} \subseteq \mathbf{FNPSPACE}$).

Here we state the reason that we regard $\mathbf{FNPSPACE}_{\text{p-out}}$ as a class that well captures the difficulty of Sokoban. First, the class naturally corresponds to **PSPACE**.

Proposition 4.2. $dom(\mathbf{FNPSPACE}) = dom(\mathbf{FNPSPACE}_{\text{p-out}}) = \mathbf{PSPACE}$.

Moreover any problem in $\mathbf{FNPSPACE}_{\text{p-out}}$ is efficiently verifiable.

Proposition 4.3. *For any multivalued function f in* $\mathbf{FNPSPACE}_{\text{p-out}}$*, the graph of f belongs to* **P**.

We present the first ASP-$\mathbf{FNPSPACE}_{\text{p-out}}$-complete problem. It is a $\mathbf{FNPSPACE}_{\text{p-out}}$ version of the Halting Problem of TMs and thus called *bh-NPSPACE*, short for 'Bounded Halting NPSPACE'.

Definition 4.3. The multivalued function *bh-NPSPACE* is defined as follows.

- Input: A tuple (M, x, s, t) such that M is an OB-transducer; x is a string; s is an integer in unary notation; t is an integer in binary notation.
- Output: A pair (y, t') such that t' is an integer in unary; y is an output of a run of $M(x)$ within space s and exactly in time t'; moreover $t' \leq t$ holds.

Theorem 4.4. *The multivalued function bh-NPSPACE is ASP-complete in* $\mathbf{FNPSPACE}_{\text{p-out}}$.

Proof. First we show that for any multivalued function f in $\mathbf{FNPSPACE}_{\text{p-out}}$ there is a polynomial-time function-parsimonious reduction $(\phi_{\mathrm{I}}, \phi_{\mathrm{F}}, \phi_{\mathrm{B}})$ from f to *bh-NPSPACE*. There must be an OB-transducer M, polynomial functions $S(\cdot)$ and $U(\cdot)$ and an exponential function $T(\cdot)$ such that M represents f, M's running time is bounded by both $T(n)$ and $U(m)$ and M's work space is bounded by $S(n)$, where n is an input length and m is an output length. Then we define ϕ_{I}, ϕ_{F}, and ϕ_{B} as follows:

- $\phi_{\mathrm{I}}(x) \stackrel{\text{def}}{=} (M, x, S(|x|), T(|x|))$;
- $\phi_{\mathrm{F}}(x, y) \stackrel{\text{def}}{=} (y, T')$, where T' is the running time of a run of $M(x)$ that outputs y; and
- $\phi_{\mathrm{B}}(x, (y, \cdot)) = y$.

(If an input to each of the functions is not well-formed then the value is defined to be an empty string instead.) It is straightforward to confirm that the tuple $(\phi_{\mathrm{I}}, \phi_{\mathrm{F}}, \phi_{\mathrm{B}})$ is a desired reduction except that ϕ_{F} is polynomial-time computable. However, $\phi_{\mathrm{F}}(x, y)$ can be computed by simulating the run of $M(x)$ that outputs y. This simulation is possible deterministically since M is an OB-transducer, and takes at most $O(T')$ time. Since $T' \leq U(|y|)$ and U is a polynomial, it is shown that ϕ_{F} is polynomial-time computable. (Note also that the length of $\phi_{\mathrm{I}}(x)$ is polynomial in $|x|$.)

Next we show that *bh-NPSPACE* itself belongs to $\mathbf{FNPSPACE}_{\text{p-out}}$. This function can be represented by an 'universal OB-transducer' U which simulates $M(x)$ for input (M, x, s, t). Since an output includes unary t' (running time), it can be said that U runs in polynomial time in output length. ∎

Since *bh-NPSPACE* is a quite generic problem, it is easy to show that the problem satifies the property (2).

Proposition 4.5.
bh-NPSPACE $\leq_{\text{f\#}}$ ASP^1-*bh-NPSPACE*.

Proof. We assume $\{0, 1\} \subseteq \Sigma$ without loss of generality. Let $\nu(\cdot)$ be defined as follows: for an OB-transducer M, $\nu(M)$ is defined as an OB-transducer that runs in either of the following ways: (i) outputs 0 and then accepts; or (ii) outputs 1 and then runs exactly the same as M. Then we define ϕ_{I}, ϕ_{F}, and ϕ_{B} as follows:

- $\phi_{\mathrm{I}}(M, x, s, t) \stackrel{\text{def}}{=} (\nu(M), x, s, t + 1)$;
- $\phi_{\mathrm{F}}(x, (y, t')) \stackrel{\text{def}}{=} (1y, t' + 1)$; and
- $\phi_{\mathrm{B}}(x, (1y, t')) = (y, t' - 1)$.

Then it is straightforward to confirm that *bh-NPSPACE* $\leq_{\text{f\#}}$ ASP^1-*bh-NPSPACE* via $(\phi_{\mathrm{I}}, \phi_{\mathrm{F}}, \phi_{\mathrm{B}})$. ∎

5 ASP-completeness of QBF Game

Definition 5.1. *QBFGame* is a multivalued function defined as follows:

- Input: A position π of QBF Game;
- Output: A trace for π. We assume that a tree is stringified in a certain way that nodes appear in preorder, without specifying further detail.

Here we show that *QBFGame* is ASP-complete in $\mathbf{FNPSPACE}_{\text{p-out}}$.

Lemma 5.1. *The function QBFGame belongs to* $\mathbf{FNPSPACE}_{\text{p-out}}$.

Proof. We construct an OB-transducer M which computes *QBFGame* as follows. For an input π (a position of QBF Game), M does the folloing recursively:

- When π is Player 1's turn ($\exists$), then M chooses a value (0 or 1) to assign to the target variable and obtains the reduced expression π'. Then M writes π' and computes for π' recursively.
- When π is Player 2's turn ($\forall$), M runs as above except that M recurses twice: it first chooses value 0 and next chooses 1.

In this way positions are arranged in the right way on the output. It is straightforward to confirm that M is polynomial space bounded. ∎

Lemma 5.2.
bh-NPSPACE $\leq_{\text{f\#}}$ *QBFGame.*

Proof. We constructs an function-parsimonious reduction $(\phi_{\mathrm{I}}, \phi_{\mathrm{F}}, \phi_{\mathrm{B}})$. It is a minor modification to the reduction given by Stockmeyer and Meyer [6] that shows **PSPACE**-completeness of QBF.

The instance reduction ϕ_{I} is given as follows. Let (M, x, s, t) be an instance of *bh-NPSPACE*. Here we only consider instances such that t is a power of two. The function ϕ_{I} maps the instance to a QBF formula in the same as in [6]. That is we construct a formula that signifies 'middle-first' reachability in the space of configurations of M as follows. Let $\mathit{reach}(A, B, T)$ denote the predicate "M can move from A to B within T steps." Then the fact "M accepts x" is expressed as $\mathit{reach}(C_1, C_2, t)$ where C_1 is the initial configuration and C_2 is the accepting configuration (computed from M, x and s). Then we recursively expand the expression by the following equality:

$$\begin{aligned}\mathit{reach}(A, B, T) \equiv \exists Z \forall X \forall Y \\ (((X{=}A \wedge Y{=}Z) \vee (X{=}Z \wedge Y{=}B)) \\ \Longrightarrow \mathit{reach}(X, Y, T/2)).\end{aligned}$$

The base case $\mathit{reach}(A, B, 1)$ can be described as expression of variables in A and B according to M's trasition function.

The solution reductions ϕ_{F} and ϕ_{B} are the natural correspondence of solutions of both problems which are derived from the same run of the transducer M. Since both strings contain sufficient information to completely restore the run of M, these reductions can be computed in polynomial time (although the input length itself can be superpolynomial with respect to the input to ϕ_{I}). ∎

Theorem 5.3. *QBFGame is ASP-complete in* $\mathbf{FNPSPACE}_{\text{p-out}}$.

Corollary 5.4. *For any integer n, the language dom(*ASP^n*-QBFGame) (that is, the decision version of n-ASP of QBFGame) is* **PSPACE***-compelte.*

6 Conclusion

We showed that *QBFGame* is ASP-complete in the class $\mathbf{FNPSPACE}_{\text{p-out}}$, in which (a three-dimension version of) Sokoban is also ASP-complete. This result suggests that the equivalence between two-player games with polynomial duration and one-player puzzles with superpolynomial duration holds in the world of the function problems.

References

[1] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28:114–133, 1981.

[2] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[3] A. L. Selman. Much ado about functions. In *Proc. Conf. on Structures in Complexity Theory*, pp. 198–212, 1996.

[4] T. Seta and T. Yato. Hardness of finding another solution to problems in PSPACE — with application to Sokoban puzzle game. submitted for publication.

[5] L. J. Stockmeyer and A. K. Chandra. Provably difficult combinatorial games. *SIAM J. Comput.*, 8(2):151–174, 1979.

[6] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In *Proceedings of the 5th ACM Symposium on Theory of Computing (STOC'73)*, pp. 1–9, 1973.

[7] N. Ueda and T. Nagao. NP-completeness results for NONOGRAM via parsimonious reductions. Technical Report TR96-0008, Department of Computer Science, Tokyo Institute of Technology, 1996.

[8] T. Yato and T. Seta. Complexity and completeness of finding another solution and its application to puzzles. *IEICE Trans. Fundamentals*, E86-A(5):1052–1060, 2003.