A Temporal Proof System for General Game Playing

Michael Thielscher
School of Computer Science and Engineering
The University of New South Wales, Australia
mit@cse.unsw.edu.au

Sebastian Voigt
Department of Computer Science
Dresden University of Technology, Germany
sebastian.voigt@inf.tu-dresden.de

Abstract

A *general* game player is a system that understands the rules of unknown games and learns to play these games well without human intervention. A major challenge for research in General Game Playing is to endow a player with the ability to extract and prove game-specific knowledge from the mere game rules. We define a formal language to express temporally extended—yet local—properties of games. We also develop a provably correct proof theory for this language using the paradigm of Answer Set Programming, and we report on experiments with a practical implementation of this proof system in combination with a successful general game player.

1 Introduction

General Game Playing is concerned with the development of systems that understand the rules of previously unknown games and learn to play these games well without human intervention. Recently identified as a Grand Challenge for Artificial Intelligence, this endeavour requires to combine methods from a variety of sub-disciplines, including reasoning, search, computer game playing, and learning (Pell 1993; Genesereth, Love, and Pell 2005). The general Game Description Language (GDL) (Genesereth, Love, and Pell 2005) has been developed for the purpose of communicating the rules of unknown n-player games ($n \geq 1$) to a general game player. GDL rules are logical axioms, and a plain, Prolog-like inference mechanism suffices for a basic general game player to be able to make legal moves (Schiffel and Thielscher 2009b). Simple games can then be solved by complete search, and recent research in General Game Playing has shown that Monte-Carlo methods provide a successful form of selective blind search to address games that are more complex (Björnsson and Finnsson 2009).

Moving from blind to informed search, however, is a great endeavour in General Game Playing as it requires a player to fully automatically analyse the bare rules of unknown games with the goal to extract and exploit game-specific knowledge. This ability to form knowledge about a new game is a prerequisite for both the automated generation of search heuristics and the construction of evaluation functions for non-terminal positions (Kuhlmann, Dresner, and Stone 2006; Clune 2007; Schiffel and Thielscher 2007). While successful General Game Playing systems like the ones just mentioned do extract this kind of knowledge, they do not actually attempt to prove it; rather they generate a number of random sample matches to test a property, and then rely on the correctness of this informed guess.

In this paper, we present the first formal yet practical approach to the formalisation and automated proving of *local* yet *temporally extended* properties of games on the basis of the Game Description Language. By "local" we mean properties which do not require to analyse the entire game tree and can be considered as invariants of reachable states.[1] By "temporally extended" we mean properties that concern two or more successive game states.[2] To this end,

- We define syntax and semantics of a formal language to express game-specific, local knowledge.
- We develop a proof theory that allows to verify these formulas against a given GDL game specification.
- We briefly report on first experiments with a practical implementation of a proof system for this theory.

For describing game-specific properties, we combine elements from GDL with Temporal Logic. Our proof theory employs the paradigm of Answer Set Programming (ASP) (see, e.g., (Gelfond 2008)) and builds on a recent and basic method for ascertaining simple *static* properties of games, that is, which hold across all positions (Schiffel and Thielscher 2009a). For the implementation, we have integrated an off-the-shelf, state-of-the-art ASP system (Potassco 2008) with a successful knowledge-based general game player (Schiffel and Thielscher 2007). Before we start, however, we should stress that in this paper we are only concerned with automatically *proving* knowledge, not with automatically *finding* properties worth proving.[3]

[1] An example of a *global* property would be the existence of a winning strategy for a player. This cannot be expressed by an invariant, as it may hold initially but not for all reachable states. Our interest here lies in games that are far too complex to enable automatic proofs of global properties in practice.

[2] A simple, concrete example is the fact that in standard Tic-Tac-Toe a marked cell persists from one position to the next.

[3] We refer to (Clune 2007; Schiffel and Thielscher 2007) for an extensive discussion on various types of game-specific knowledge that helps a general game player find good heuristics and generate tailor-made evaluation functions.

```
role(xplayer).  role(oplayer).
init(control(xplayer)).  init(cell(1,1,b)).  ...  init(cell(3,3,b)).

legal(P,mark(X,Y))  :- true(control(P)), true(cell(X,Y,b)).
legal(xplayer,noop) :- true(control(oplayer)).
legal(oplayer,noop) :- true(control(xplayer)).

next(cell(M,N,x)) :- does(xplayer,mark(M,N)).
next(cell(M,N,o)) :- does(oplayer,mark(M,N)).
next(cell(M,N,W)) :- true(cell(M,N,W)), distinct(W,b).
next(cell(M,N,b)) :- true(cell(M,N,b)), does(P,mark(I,J)), distinct(M,I).
next(cell(M,N,b)) :- true(cell(M,N,b)), does(P,mark(I,J)), distinct(N,J).
next(control(oplayer)) :- true(control(xplayer)).
next(control(xplayer)) :- true(control(oplayer)).
```

Figure 1: A GDL description of Tic-Tac-Toe (without the definition of termination and goalhood). A game position is encoded using features *control*(P), where $P \in \{$*xplayer*, *oplayer*$\}$, and *cell*(X, Y, C), where $X, Y \in \{1, 2, 3\}$ and $C \in \{x, o, b\}$.

2 Game Description Language

The Game Description Language (GDL) has been developed to formalise the rules of any finite game with complete information in such a way that the description can be automatically processed by a general game player. Due to lack of space, we can give just a very brief introduction to GDL and have to refer to (Love et al. 2006) for details.

GDL is based on the standard syntax of logic programs, including negation. We assume familiarity with the basic notions of logic programming. We adopt the Prolog convention according to which variables are denoted by uppercase letters and predicate and function symbols start with a lowercase letter. As a tailor-made specification language, GDL uses a few pre-defined predicate symbols:

`role(R)`	`R` is a player
`init(F)`	`F` holds in the initial position
`true(F)`	`F` holds in the current position
`legal(R,M)`	player `R` has legal move `M`
`does(R,M)`	player `R` does move `M`
`next(F)`	`F` holds in the next position
`terminal`	the current position is terminal
`goal(R,N)`	player `R` gets goal value `N`

A further standard predicate is `distinct(X,Y)`, which means syntactic inequality of the two arguments. GDL imposes restrictions on the use of these keywords:

- `role` only appears in facts;
- `init` and `next` only appear as head of clauses, and `init` does not depend on any of `true`, `legal`, `does`, `next`, `terminal`, or `goal`;
- `true` and `does` only appear in clause bodies with `does` not depending on `legal`, `terminal`, or `goal`.

As an example, Figure 1 shows an excerpt of a GDL description for the simple game of Tic-Tac-Toe. GDL imposes some further, general restrictions on a set of clauses with the intention to ensure finiteness of the set of derivable predicate instances. Specifically, the program must be *stratified* (Apt, Blair, and Walker 1987) and *allowed* (Lloyd and Topor 1986). Stratified logic programs are known to admit a specific *standard model* (Apt, Blair, and Walker 1987).

Based on the concept of the standard model, a GDL description can be understood as a state transition system as follows. To begin with, any valid game description G in GDL contains a finite set of function symbols, including constants, which implicitly determines a set of ground terms Σ. This set constitutes the symbol base Σ in the formal semantics for G.

The players and the initial position of a game can be directly determined from the clauses for, respectively, `role` and `init` in G. In order to determine the legal moves, update, termination, and goalhood for any given position, this position has to be encoded first, using the keyword `true`. To this end, for any *finite* subset $S = \{f_1, \ldots, f_n\} \subseteq \Sigma$ of a set of ground terms, the following set of logic program facts encodes S as the current position:

$$S^{\texttt{true}} \stackrel{\text{def}}{=} \{\texttt{true}(f_1)\texttt{.}, \ \ldots, \ \texttt{true}(f_n)\texttt{.}\}$$

Furthermore, for any function $A : (\{r_1, \ldots, r_k\} \mapsto \Sigma)$ that assigns a move to each player $r_1, \ldots, r_k \in \Sigma$, the following set of facts encodes A as a joint move:

$$A^{\texttt{does}} \stackrel{\text{def}}{=} \{\texttt{does}(r_1, A(r_1))\texttt{.}, \ \ldots, \ \texttt{does}(r_k, A(r_k))\texttt{.}\}$$

Definition 1 *Let G be a GDL specification whose signature determines the set of ground terms Σ. Let 2^Σ be the set of* finite *subsets of Σ. The* semantics *of G is the state transition system $(R, S_{init}, T, l, u, g)$ where*[4]

- $R = \{r \in \Sigma : G \vDash \texttt{role}(r)\}$ *(the players);*
- $S_{init} = \{f \in \Sigma : G \vDash \texttt{init}(f)\}$ *(the initial position);*
- $T = \{S \in 2^\Sigma : G \cup S^{\texttt{true}} \vDash \texttt{terminal}\}$ *(the terminal positions);*
- $l = \{(r, a, S) : G \cup S^{\texttt{true}} \vDash \texttt{legal}(r, a)\}$, *where* $r \in R$, $a \in \Sigma$, *and* $S \in 2^\Sigma$ *(the legality relation);*
- $u(A, S) = \{f \in \Sigma : G \cup S^{\texttt{true}} \cup A^{\texttt{does}} \vDash \texttt{next}(f)\}$, *for all* $A : (R \mapsto \Sigma)$ *and* $S \in 2^\Sigma$ *(the update function);*
- $g = \{(r, v, S) : G \cup S^{\texttt{true}} \vDash \texttt{goal}(r, v)\}$, *where* $r \in R$, $v \in \mathbb{N}$, *and* $S \in 2^\Sigma$ *(the goal relation).*

[4] Below, entailment $\vDash$ is via the aforementioned standard model for a set of clauses.

For $S, S' \in 2^\Sigma$ we write $S \xrightarrow{A} S'$ if $A : (R \mapsto \Sigma)$ is such that $(r, A(r), S) \in l$ for each $r \in R$ and $S' = u(A, S)$ (and $S \notin T$). We call $S_0 \xrightarrow{A_0} S_1 \xrightarrow{A_1} \ldots \xrightarrow{A_{m-1}} S_m$ (where $m \geq 0$) a sequence *(of legal moves), sometimes abbreviated as $(S_0, S_1, \ldots, S_m)$.*

This definition provides a formal semantics by which a GDL description is interpreted as an abstract k-player game: in every position S, starting with S_{init}, each player r chooses a move a that satisfies $l(r, a, S)$. As a consequence the game state changes to $u(A, S)$, where A is the joint move. The game ends if a position in T is reached, and then g determines the outcome. The restrictions in GDL ensure that entailment wrt. the standard model is decidable and that only finitely many instances of each predicate are entailed. This guarantees that the definition of the semantics is effective.

3 Formalising Temporal Game Knowledge

In this section we define a formal language which allows the formulation of temporally extended yet local properties of a game given in GDL. A simple and elegant way to obtain such a language is by extending GDL by the unary operator "$\bigcirc$" borrowed from Temporal Logic and used to refer to a successor game state.

Definition 2 *Let G be a GDL description and $\mathcal{P}$ be the set of ground atoms $p(\vec{t})$ over the signature of G such that $p \notin \{\texttt{init}, \texttt{next}\}$ and p does not depend on* `does` *in G. We define the set F of* formulas *to be the smallest set such that $\mathcal{P} \subseteq F$ and F is closed under $\neg$, $\wedge$ and $\bigcirc$.*

We also define over syntax trees t_φ and t_ψ of $\varphi, \psi \in F$:

- *deg(φ) (the* degree of *φ): the maximal number of occurrences of $\bigcirc$ on paths from the root of t_φ to its leaves;*
- *lev(ψ, φ) (the* levels of *ψ* wrt. *φ): a set of integers such that $i \in$ lev(ψ, φ) iff t_ψ is a subtree of t_φ and there are i occurrences of $\bigcirc$ on the path from the root of t_φ to the root of t_ψ.*

We define $\vee$ and $\supset$ as the usual macros and use restricted quantification $\forall(X : X_{dom})\,\varphi$ to abbreviate $\bigwedge_{t \in X_{dom}} \varphi[X/t]$ for *finite* subsets X_{dom} of Σ. We also allow counting quantifiers of the form $\exists_{m..n}(X : X_{dom})\,\varphi$ to formulate that there are at least m and at most n instances for X for which φ is true. Modality $\bigcirc\varphi$ states that φ holds in all positions that are a direct, legal successor of the current game state.As an example for a formula, consider the Tic-Tac-Toe property that once the cell at position $(1,1)$ has been marked by *xplayer*, it will keep this mark in every legal successor state. This can be formulated as $true(cell(1,1,x)) \supset \bigcirc true(cell(1,1,x))$. The degree of this formula is 1 and the levels of subformula $true(cell(1,1,x))$ are $\{0, 1\}$.

Semantics Intuitively, a formula should be true in a state S only if it is satisfied by all "relevant" sequences starting at S. Clearly, sequences of length greater than $n = deg(\varphi)$ need not be considered: Since future states that are more than n steps away carry no information about the formula, these sequences can be reduced to their initial subsequences of length n. Sequences shorter than n, however, must be taken into account. Otherwise, a formula like $\psi \wedge \bigcirc\rho$ would be considered true in each terminal state S_t regardless of the truth of ψ, as no legal sequence of length ≥ 1 exists in S_t. In general, a sequence shorter than n is relevant if and only if it ends in a terminal state. These considerations motivate the following definition.

Definition 3 *A sequence $(S_0, \ldots, S_m)$ is called n-*maximal *iff either $m = n$ or $m < n$ and $S_m \in T$.*

Entailment of a formula φ wrt. a state S can now be formally defined over all *deg*(φ)-maximal sequences starting at S.

Definition 4 *Let G be a GDL description, S_0 a state, and φ a formula such that deg(φ) $= n$. We say that S_0* satisfies *φ (written $S_0 \vDash_t \varphi$) if for all n-maximal sequences $S_0 \xrightarrow{A_0} \ldots \xrightarrow{A_{m-1}} S_m$ ($m \leq n$) we have that $(S_0, \ldots, S_m) \vDash_t \varphi$ according to the following definition:*

$$
\begin{array}{lll}
(S_i, \ldots, S_m) \vDash_t p & \textit{iff} & G \cup S_i^{\texttt{true}} \vDash p \quad (p \in \mathcal{P}) \\
(S_i, \ldots, S_m) \vDash_t \neg\varphi & \textit{iff} & (S_i, \ldots, S_m) \nvDash_t \varphi \\
(S_i, \ldots, S_m) \vDash_t \varphi_1 \wedge \varphi_2 & \textit{iff} & (S_i, \ldots, S_m) \vDash_t \varphi_1 \textit{ and } \\
 & & (S_i, \ldots, S_m) \vDash_t \varphi_2 \\
(S_i, \ldots, S_m) \vDash_t \bigcirc\varphi & \textit{iff} & (S_{i+1}, \ldots, S_m) \vDash_t \varphi \quad (i < m) \\
(S_m) \vDash_t \bigcirc\varphi & &
\end{array}
$$

A crucial part here is $(S_m) \vDash_t \bigcirc\varphi$: In case we reach the end of a state sequence, every formula of the form $\bigcirc\varphi$ must be true. Note that this implies $\bigcirc\varphi$ to be true in every terminal state even if φ is inconsistent. In our setting this is perfectly acceptable as we are just interested in the truth of a formula in reachable states—all states beyond are irrelevant. It is also worth mentioning that, therefore, $\neg\bigcirc\varphi$ implies $\bigcirc\neg\varphi$ but not vice versa.

4 Encoding Temporal Game Knowledge

In the following we first present an encoding of temporal formulas as logic program clauses which, together with a set of GDL rules, will then enable us to define a suitable proof method. Since we consider properties that are local but may involve sequences of successive game states, we first need to define the *temporal extension* (with a horizon n) of a given GDL specification.

Definition 5 *Let G be a GDL description and G_{init} the set of all clauses of G with head* `init`*. For $n \geq 0$ we define $G_n = \bigcup_{0 \leq i \leq n} \{c^i \mid c \in (G \setminus G_{init})\}$, where $\cdot^i$ replaces each occurrence of*

- `next`(f) *by* `true`$(f, i+1)$ *and*
- $p(t_1, \ldots, t_n)$ *by* $p(t_1, \ldots, t_n, i)$, *if* $p \notin \{\texttt{role}, \texttt{distinct}, \texttt{next}\}$.

Extending the definitions of $S^{\texttt{true}}$ and $A^{\texttt{does}}$ (cf. Section 2), we define their timed variants as

$$S^{\texttt{true}}(0) = \{\texttt{true}(f, 0)\,. \mid f \in S\} \textit{ and}$$

$$A^{\texttt{does}}(i) = \{\texttt{does}(r_1, A(r_1), i), \ldots, \texttt{does}(r_k, A(r_k), i)\}$$

As an example, consider the fourth rule with head `next` in the GDL description G of Figure 1. Its temporal extension G_n contains a temporally extended rule for every $i \leq n$; e.g. the following clause for $i = 0$.

```
true(cell(M,N,b),1) :-
   true(cell(M,N,b),0),
   does(P,mark(I,J),0), distinct(M,I).
```

Note that G_n could easily be defined such that it is stratified: instead of extending predicates p by a time argument, time could be encoded into their names, obtaining different predicates p_i for each time step. We find Definition 5 more convenient but will nonetheless assume G_n to be stratified. This assumption is needed for the following result, which shows that a temporally extended GDL description can be used to reason about limited sequences of state transitions.

Theorem 1 Consider a GDL description G and a sequence $S_0 \stackrel{A_0}{\rightarrow} S_1 \ldots \stackrel{A_{m-1}}{\rightarrow} S_m$. Let $P = S_0^{\mathtt{true}}(0) \cup G_m \cup \bigcup_{i=0}^{m-1} A_i^{\mathtt{does}}(i)$, then for all $0 \leq i \leq m$ and predicate symbols $p \notin \{\mathtt{init}, \mathtt{next}\}$ that do not depend on `does` we have

- $S_i = \{f \mid P \vDash \mathtt{true}(f,i)\}$
- $G \cup S_i^{\mathtt{true}} \vDash p(\vec{t})$ iff $P \vDash p(\vec{t},i)$

Proof (sketch): By induction on m. The main argument is the existence of a stratification of P such that for each timepoint i the rules concerning i lie in a lower stratum than the rules concerning timepoint $i+1$. □

Since predicate `legal` never depends on `does` in valid GDL descriptions, the second item of Theorem 1 together with Definition 5 implies $P \vDash \mathtt{legal}(r, A_i(r), i)$ for all $r \in R$ and $0 \leq i \leq m-1$. Similarly, $P \vDash \mathtt{terminal}(m)$ holds if and only if S_m is a terminal state.

Next we show how game-specific knowledge in form of temporal formulas can be encoded as logic programs. Specifically, we define the mapping of every subformula ψ of a formula $\varphi \in F$ to a logic program relative to a natural number i which indicates the level of ψ wrt. φ. The definition assumes a function η such that $\eta(\psi, i)$ gives an atom of arity 0 which is unique for every ψ and i and not used elsewhere.

Definition 6 *Let $\psi \in F$ and $i \in \mathbb{N}$. The* encoding of ψ at level i*, denoted $\mathit{Enc}(\psi, i)$, is recursively defined:*

$$
\begin{array}{lll}
\mathit{Enc}(p(\vec{t}),i) & = & \{\eta(p(\vec{t}),i) \texttt{ :- } p(\vec{t},i)\texttt{.}\} \\
\mathit{Enc}(\neg\psi,i) & = & \{\eta(\neg\psi,i) \texttt{ :- } \neg\, \eta(\psi,i)\texttt{.}\} \cup \mathit{Enc}(\psi,i) \\
\mathit{Enc}(\psi_1 \wedge \psi_2,i) & = & \{\eta(\psi_1 \wedge \psi_2,i) \texttt{ :- } \eta(\psi_1,i), \eta(\psi_2,i)\texttt{.}\} \\
 & & \quad \cup\ \mathit{Enc}(\psi_1,i) \ \cup\ \mathit{Enc}(\psi_2,i) \\
\mathit{Enc}(\bigcirc\psi,i) & = & \{\eta(\bigcirc\psi,i) \texttt{ :- } \mathtt{terminal}(i)\texttt{.}, \\
 & & \quad \eta(\bigcirc\psi,i) \texttt{ :- } \eta(\psi,i+1)\texttt{.}\} \\
 & & \quad \cup\ \mathit{Enc}(\psi,i+1)
\end{array}
$$

Put in words, a predicate $p(\vec{t})$ at level i is translated to a rule which entails $\eta(p(\vec{t}),i)$ if $p(\vec{t},i)$ holds. Formulas with connectives $\neg$, $\wedge$, and $\bigcirc$ recursively resolve to their correspondent subformulas. Note that $\mathit{Enc}(\bigcirc\psi,i)$ entails $\eta(\bigcirc\psi,i)$ in case level i is terminal or subformula ψ is true at level $i+1$. As an example, recall from above the formula for Tic-Tac-Toe, $\mathit{true}(\mathit{cell}(1,1,x)) \supset \bigcirc \mathit{true}(\mathit{cell}(1,1,x))$. Macro expansion results in $\neg(\mathit{true}(\mathit{cell}(1,1,x)) \wedge \neg\bigcirc \mathit{true}(\mathit{cell}(1,1,x)))$ by means of double negation, and hence the following encoding.

```
a1 :- not a2.     a2 :- a3, a4.
a3 :- true(cell(1,1,x),0).
a4 :- not a5.     a5 :- terminal(0).
a5 :- a6.   a6 :- true(cell(1,1,x),1).
```

It is easy to see that the size of the encoding of a given formula is always linear in the size of the original formula. Together with the underlying temporally extended GDL description the given encoding is correct wrt. the definition of formula entailment, as the following result shows.

Theorem 2 Let φ be a formula s.t. $\mathit{deg}(\varphi) = n$, G be a GDL description, $S_0 \stackrel{A_0}{\rightarrow} S_1 \ldots \stackrel{A_{m-1}}{\rightarrow} S_m$ be n-maximal, and $P = S_0^{\mathtt{true}}(0) \cup G_n \cup \bigcup_{i=0}^{m-1} A_i^{\mathtt{does}}(i) \cup \mathit{Enc}(\varphi, 0)$. Then $(S_0, \ldots, S_m) \vDash_t \varphi$ iff $P \vDash \eta(\varphi, 0)$.

Proof: Let ψ be a subformula of φ. We prove by induction on the structure of ψ that for all $l \in \mathit{lev}(\psi, \varphi)$ s.t. $l \leq m$: $(S_l, \ldots, S_m) \vDash_t \psi$ iff $P \vDash \eta(\psi, l)$. Note that we have $\mathit{Enc}(\psi, l) \subseteq \mathit{Enc}(\varphi, 0)$ and hence P contains all clauses referred to in the proof.

Base case $\psi = p(\vec{t})$ holds by Th.1; cases $\psi = \neg\rho$ and $\psi = \rho_1 \wedge \rho_2$ follow immediately by the induction hypothesis (IH). Now consider $\psi = \bigcirc\rho$. Case $l < m$ follows by IH applied to sequence $(S_{l+1}, \ldots, S_m)$. Case $l = m$ yields $(S_m) \vDash_t \bigcirc\rho$. Moreover, either $m = n$ or both $m < n$ and $S_m \in T$ (since the sequence is n-maximal). From $m = n$ it follows that $(n+1) \in \mathit{lev}(\rho, \varphi)$, which contradicts $\mathit{deg}(\varphi) = n$. Hence $m < n$ and $S_m \in T$, which gives (by Th.1) $P \vDash \mathtt{terminal}(m)$ and thus $P \vDash \eta(\bigcirc\rho, m)$. □

5 Proving Temporal Game Knowledge

For a general game player, showing the correctness of a given property by complete search through the state transition diagram for a game, as in (van der Hoek, Ruan, and Wooldridge 2007), is not practically feasible. To overcome this, Schiffel and Thielscher (2009a) suggested a local proof method based on Answer Set Programming (ASP) to verify simple static properties for all finitely reachable states in a GDL game. In the following, we generalise their basic idea to obtain a local proof method for temporally extended properties. *Answer sets* form a specific class of models of logic programs with negation (for details, see e.g. (Gelfond 2008)). In the following, we use two common additions that have been defined for ASP (Niemelä, Simons, and Soininen 1999): a *weight atom* $m\ \{\ p : d(\vec{x})\ \}\ n$ means that for atom p an answer set has at least m and at most n different instances that satisfy $d(\vec{x})$. Both m and n can be omitted, in which case there is no lower (respectively, upper) bound. A *constraint* is a rule `:-` $b_1, \ldots, b_k$, which excludes any answer set that satisfies $b_1, \ldots, b_k$.

To prove that a temporal formula φ holds in each reachable state S (i.e., $S \vDash_t \varphi$), we will construct two answer set programs dependent on φ in order to establish proofs for a base case and an induction step. The base case shows that φ is entailed in the initial state. The induction step shows that, provided a state entails φ, each legal successor state will also entail φ. In conclusion, then, φ is entailed in all reachable states. We assume a set of negation-free clauses $\mathcal{D}_n$ which defines the domains of features (`fdom`), actions

(adom), and time points $0, 1, \ldots, n$ (tdom). The encoding of each player performing a legal move in each nonterminal state is given by the following ASP clauses P_{legal}:

(1): `terminated(T) :- terminal(T).`
(2): `terminated(T+1) :- terminated(T).`
(3): `1{does(R,A,T):adom(A)}1 :- role(R), tdom(T), not terminated(T).`
(4): `:- does(R,A,T), not legal(R,A,T).`

For a GDL description G and formula φ with degree n, the answer set program for the *base case* is then defined as

$$P^{bc}_{\varphi}(G) = S^{\mathtt{true}}_{init}(0) \cup G_n \cup P_{legal} \cup \mathcal{D}_{n-1} \cup Enc(\varphi, 0) \cup \{\ \mathtt{:-}\ \eta(\varphi, 0)\,\mathtt{.}\}.$$

Hence $P^{bc}_{\varphi}(G)$ consists of an encoding for the initial state, $S^{\mathtt{true}}_{init}(0)$; a temporal GDL description up to time step n, G_n; the necessary requirements concerning legal moves together with the necessary domain descriptions, $P_{legal} \cup \mathcal{D}_{n-1}$; an encoding for the formula in the initial time step, $Enc(\varphi, 0)$; and the statement that φ should not be entailed in any model of $P^{bc}_{\varphi}(G)$, $\{\ \mathtt{:-}\ \eta(\varphi, 0)\,\mathtt{.}\}$. In case $P^{bc}_{\varphi}(G)$ has no model, the last rule implies that there is no state sequence starting at S_{init} that makes φ false—which means that φ is entailed by S_{init}.

For the *induction step* the answer set program is

$$P^{is}_{\varphi}(G) = \{(5)\} \cup G_{n+1} \cup P_{legal} \cup \mathcal{D}_n \cup Enc(\varphi, 0) \cup Enc(\bigcirc\varphi, 0) \cup \{(6), (7)\},$$

where

(5): `0{true(F,0):fdom(F)}.`
(6): `:- not` $\eta(\varphi, 0)$.
(7): `:-` $\eta(\bigcirc\varphi, 0)$.

$P^{is}_{\varphi}(G)$ deviates from $P^{bc}_{\varphi}(G)$ in that the state encoding (5) considers arbitrary states instead of the initial state. Moreover, the maximal time step is increased by one, φ is assumed to be true by (6), and $\bigcirc\varphi$ is assumed to be false by (7). Assuming that φ is entailed in S, $P^{is}_{\varphi}(G)$ not having a model implies that φ is entailed by all states S' that are direct successors of S.

As an example, recall again $\varphi = true(cell(1,1,x)) \supset \bigcirc true(cell(1,1,x))$. This formula can now be proved to hold in all reachable states: Let G be the rules in Figure 1. Answer sets for $P^{bc}_{\varphi}(G)$ do not satisfy $\eta(\varphi, 0)$ and hence must satisfy the encoded premise of φ in the initial state, `true(cell(1,1,x),0)`. This however contradicts the initial state encoding, so $P^{bc}_{\varphi}(G)$ has no answer set. Similarly, constraint (7) of $P^{is}_{\varphi}(G)$ only permits answer sets which satisfy `true(cell(1,1,x),1)` and do not satisfy `true(cell(1,1,x),2)`, contradicting the temporal extension of the third `next` rule in Figure 1.

6 Correctness of the Proof Method

We are now ready to state and prove our main result: the correctness of the proof method.

Theorem 3 Let φ be a formula for a game with GDL description G and whose initial state is S_{init}. If $P^{bc}_{\varphi}(G)$ and $P^{is}_{\varphi}(G)$ are inconsistent, then for all finite sequences $S_{init} \xrightarrow{A_0} S_1 \ldots \xrightarrow{A_{k-1}} S_k$ we have $S_k \models_t \varphi$.

Proof: Let $deg(\varphi) = n$. The proof is via induction on k.

Base Case $k = 0$: We prove that if $S_{init} \not\models_t \varphi$ then $P^{bc}_{\varphi}(G)$ admits an answer set. $S_{init} \not\models_t \varphi$ implies that there is an n-maximal sequence $S_0 \xrightarrow{A_0} S_1 \ldots \xrightarrow{A_{m-1}} S_m$ s.t. $S_{init} = S_0$ and $(S_0, \ldots, S_m) \not\models_t \varphi$. Let $\mathcal{M}$ be the standard model of $P = [P^{bc}_{\varphi}(G) \cup \bigcup_{i=0}^{m-1} A_i^{\mathtt{does}}(i)] \setminus \{(3), (4),\ \mathtt{:-}\ \eta(\varphi, 0)\,\mathtt{.}\}$. As stratified programs are known to admit a unique answer set that coincides with its standard model (Gelfond 2008), $\mathcal{M}$ is also an answer set for P. $(S_0, \ldots, S_m) \not\models_t \varphi$ implies (by Th.2) that $P \not\models \eta(\varphi, 0)$ and hence that $\mathcal{M}$ is also an answer set for $P_c = P \cup \{\ \mathtt{:-}\ \eta(\varphi, 0)\,\mathtt{.}\}$.

In the following we argue that $\mathcal{M}$ is also a model of $P^{bc}_{\varphi}(G)$. Consider S_i s.t. $i < m$: There is exactly one a for every r such that $\mathcal{M} \models \mathtt{does}(r, a, i)$, namely $a = A_i(r)$; this fulfils (3). Since $\mathcal{M} \models \mathtt{legal}(r, A_i(r), i)$ (by Th.1) we also have (4). Now consider S_m: If S_m is terminal then (by Th.1) $P_c \models \mathtt{terminal}(m)$, hence $P_c \models \mathtt{terminated}(j)$ for all $m \leq j \leq n$, which implies (3) and (4). If S_m is nonterminal then $m = n$, hence (3) and (4) are fulfilled as $\mathtt{tdom}(n)$ is false.

Induction Step: By IH we have $S_k \models_t \varphi$ and $S_k \xrightarrow{A_k} S_{k+1}$ for some A_k and S_{k+1}. We prove that if $S_{k+1} \not\models_t \varphi$, then $P^{is}_{\varphi}(G)$ admits an answer set. $S_{k+1} \not\models_t \varphi$ implies that there is an n-maximal sequence $S_{k+1} \xrightarrow{A_{k+1}} S_{k+2} \xrightarrow{A_{k+2}} \ldots \xrightarrow{A_{k+m}} S_{k+m+1}$ (where $0 \leq m \leq n$) s.t. $(S_{k+1}, \ldots, S_{k+m+1}) \not\models_t \varphi$. It follows that $(S_k, S_{k+1}, \ldots, S_{k+m+1}) \not\models_t \bigcirc\varphi$ and that $(S_k, S_{k+1}, \ldots, S_{k+m+1})$ is $(n+1)$-maximal. Let $\mathcal{M}$ be the standard model for $P = [P^{is}_{\varphi}(G) \cup \bigcup_{i=k}^{k+m} A_i^{\mathtt{does}}(i) \cup S^{\mathtt{true}}_k(0)] \setminus \{(3), (4), (5), (6), (7)\}$.

$(S_k, \ldots, S_{k+m}, S_{k+m+1}) \not\models_t \bigcirc\varphi$ gives (Th.2) $P \not\models \eta(\bigcirc\varphi, 0)$ which fulfils (7). For (6): Case $m + 1 = deg(\bigcirc\varphi)$ yields $m = n$, hence $(S_k, \ldots, S_{k+m})$ is n-maximal. Case $m+1 < deg(\bigcirc\varphi)$ and $S_{k+m+1} \in T$ gives $m+1 \leq n$, hence $(S_k, \ldots, S_{k+m}, S_{k+m+1})$ is n-maximal. Both cases imply (Th.2) $P \models \eta(\varphi, 0)$, thus satisfying (6). Now $\mathcal{M}$ remains a model by replacing $S^{\mathtt{true}}_k(0)$ with (5). The remaining argumentation is similar to the base case and implies that $\mathcal{M}$ is also a model of $P^{is}_{\varphi}(G)$. □

7 Experimental Results

We have implemented our proof method for temporally extended properties of games using Fluxplayer (Schiffel and Thielscher 2007) for the generation of the logic program, which is then passed to the grounder Bingo (Potassco 2008), in turn passing the result to the ASP solver Clasp (Potassco 2008). The domains for variables that occur in formulas as well as the domains for players (P_{dom}), moves (M_{dom}), etc. are calculated based on dependency graphs as described in (Schiffel and Thielscher 2009a), but with some optimisations (which together with some improvements concerning clauses (3) and (5) are beyond the scope of this paper). We ran tests on a number of games from previous GGP compe-

	Turn Taking	Persistence	Control
3pttc	yes,0.44	no,0.33 (*blue*)	yes,0.42
b-tictactoe	no,0.07	yes,0.18 ($\neg b$)	no,0.21
connect4	yes,0.14	yes,0.18 (*r*)	yes,0.20
endgame	yes,5.15	no,1.42 ($\neg wk$)	yes,14.75
othello	yes,2.03	no,0.89 (*red*)	yes,2.84
tictactoe	yes,0.12	yes,0.13 ($\neg b$)	yes,0.12
tttcc4	yes,7.52	no,0.56 (*redpawn*)	yes,3.02

Figure 2: Results of three selected properties ("yes" means proved true) for seven games (see *general-game-playing.de*) and respective times to prove in seconds. Experiments were run on an Intel Core 2 Duo cpu with 3.16 GHz. Persistence is for feature *cell* with ground third argument, e.g. in tictactoe $\neg b$ means that we proved $\forall(X : X_{dom})\,\forall(Y : Y_{dom})\,(\neg true(cell(X, Y, b)) \supset \bigcirc\neg true(cell(X, Y, b)))$.

titions (Genesereth, Love, and Pell 2005) to automatically verify some very common properties that are often crucial for a general game player to know. A selection is shown in Figure 2, where we exemplarily had the player try to prove

- *Turn Taking:* Always at most one player has two or more legal moves:
  $\exists_{0..1}(P : P_{dom})\,\exists_{2..\infty}(M : M_{dom})\,legal(P, M)$.
- *Persistence:* A feature f stays true [false] once it becomes true [false]: $[\neg]true(f) \supset \bigcirc[\neg]true(f)$.
- *Control:* In an n-player game, a player who has control will also have control after n moves:
  $\forall(P : P_{dom})\,(true(control(P)) \supset \bigcirc^n true(control(P)))$.

The size of the ground answer set program for a property is crucial for the proof time and has been reduced by omitting irrelevant rules, resulting in excellent computation times for many games and thus enabling various proof attempts in the practical setting of the GGP competitions, with strictly limited time to analyse a given description. Beyond the typically tight time limits in a GGP competition, the results are promising for the aim of discovering increasingly complex and interesting properties of a new game on its own right.

8 Summary

We have defined syntax and semantics of a formal language to describe game-specific properties in the context of General Game Playing. We have shown how these formulas can be encoded as a logic program on the basis of a temporal extension of GDL rules, and we have developed (and formally verified) a proof theory with the help of Answer Set Programming. While the main focus of this paper is theoretical, initial experimental results support our expectation that knowledge-based general game playing systems can make practical use of our proof method to automatically verify game-specific knowledge against a previously unknown game description. Of course our method also supports the design of new games by allowing the designer to verify that her game rules satisfy desired properties.

In terms of related work, our proof theory builds on, and significantly extends, a recent basic method for ascertaining simple static properties of games (Schiffel and Thielscher 2009a). Our semantics is inspired by work on control knowledge in planning problems (Bacchus and Kabanza 2000), adapting planning actions with preconditions and effects to joint moves with legality and update. The first method of automatically proving temporally extended properties for general games is presented in (van der Hoek, Ruan, and Wooldridge 2007), but this requires to systematically search the entire set of reachable positions in a game and therefore cannot be used in practice except for very simple games.

References

Apt, K.; Blair, H. A.; and Walker, A. 1987. Towards a theory of declarative knowledge. In *Foundations of Deductive Databases and Logic Programming*, 89–148.

Bacchus, F., and Kabanza, F. 2000. Using Temporal Logics to Express Search Control Knowledge for Planning. *Artificial Intelligence* 116(1-2):123–191.

Björnsson, Y., and Finnsson, H. 2009. CADIAPLAYER: A simulation-based general game player. *IEEE Transactions on Computational Intelligence and AI in Games* 1(1):4–15.

Clune, J. 2007. Heuristic evaluation functions for general game playing. In *AAAI*, 1134–1139.

Gelfond, M. 2008. Answer sets. In *Handbook of Knowledge Representation*, 285–316. Elsevier.

Genesereth, M.; Love, N.; and Pell, B. 2005. General game playing: Overview of the AAAI competition. *AI Magazine* 26(2):62–72.

Kuhlmann, G.; Dresner, K.; and Stone, P. 2006. Automatic heuristic construction in a complete general game player. In *AAAI*, 1457–1462.

Lloyd, J., and Topor, R. 1986. A basis for deductive database systems II. *J. of Logic Programming* 3(1):55–67.

Love, N.; Hinrichs, T.; Haley, D.; Schkufza, E.; and Genesereth, M. 2006. General Game Playing: Game Description Language Specification. Technical Report LG–2006–01, Stanford University. Available at *games.stanford.edu*.

Niemelä, I.; Simons, P.; and Soininen, T. 1999. Stable model semantics of weight constraint rules. In *Proceedings of LPNMR*, vol. 1730 of *LNCS*, 317–331.

Pell, B. 1993. *Strategy Generation and Evaluation for Meta-Game Playing*. Ph.D., Cambridge.

Potassco 2008. Potsdam Answer Set Solving Collection. Available at *potassco.sourceforge.net/*.

Schiffel, S., and Thielscher, M. 2007. Fluxplayer: A successful general game player. In *AAAI*, 1191–1196.

Schiffel, S., and Thielscher, M. 2009a. Automated theorem proving for general game playing. In *IJCAI*, 911–916.

Schiffel, S., and Thielscher, M. 2009b. A multiagent semantics for the Game Description Language. In *Agents and Artificial Intelligence: Proceedings of ICAART*, vol. 67 of *Communications in Computer and Information Science* Springer.

van der Hoek, W.; Ruan, J.; and Wooldridge, M. 2007. Strategy logics and the game description language. In *Proceedings of the Workshop on Logic, Rationality and Interaction*. Beijing.