Abstract interpretations of games

Perdita Stevens [⋆]

Laboratory for Foundations of Computer Science
Division of Informatics, University of Edinburgh
The King's Buildings, Mayfield Road
Edinburgh EH9 3JZ
Scotland,UK

Abstract. In recent work [20] we have proposed *abstract games*, and an example algorithm for finding winning strategies of them, as an approach to verification problems – such as model-checking and equivalence checking – which permits both a variable level of abstraction and on-the-fly exploration. It was clear that this work had something in common with abstract interpretation. This paper, which describes work in progress, generalises and makes the connection explicit.

1 Introduction

In computer science questions which we need to answer often refer to objects which are, in an appropriate sense, infinite state. Examples include value-passing processes over infinite data domains, and timed automata, where the source of infiniteness is the fact that clocks may show any real-numbered time. To answer questions about such objects we obviously need techniques other than exhaustive exploration. (In concurrency we may be interested, for example, in the question of whether a system satisfies a formula in some logic (model-checking) or in whether some pair of processes is in some relation (equivalence checking, preorder checking, etc.)) The standard approach is to define some finite representation of the infinite object, detailed enough to answer all questions from some class about the object, and to prove that the answers obtained using it are right, that is, that the abstraction is sound for the class of questions considered. For example, in the case of timed automata, [1] showed that we may work with a finite *region automaton*. An early approach to value-passing process algebras was [14], which dealt with bisimulation of data-independent processes. Later *symbolic transition graphs* (STGs) [13] and then *symbolic transition graphs with assignment* (STGAs) [15] were introduced to handle a larger class of value-passing processes. This representation has been used to answer bisimulation questions and also model-checking questions [18]. This approach also includes many explicitly abstract interpretation (a.i.) based papers, such as [12], [7], [8], [9], and many others.

This approach has clearly proved useful, but there are at least two possible problems with it. First, the correctness, with respect to the class of questions of interest, of the particular finite representation has to be proved in each case, although the work is often routine. Secondly, and more seriously, there often is no finite representation suitable for answering all questions from a well-defined class. We would prefer a tool not to refuse to answer easy questions because some questions about the system happen to be hard. This problem is acute with verification problems: the easiness of a question is a property of the question, not of the system about which it is asked. Am infinite-state system may, for example, have no finite state system which satisfies all the same properties (from some class), and yet a given property in that class may be "easy" to verify.

In recent years the study of games as an approach to problems in concurrency has taken off. It has long been recognised that bisimilarity, and many other equivalences, can be characterised by games reminiscent of Ehrenfeucht-Fraissé games; see, for example, [22]. Very briefly, we can define a bisimulation game on a pair of processes (E, F) by saying that the two players (Abelard and Eloise) move alternately. Abelard, who starts, chooses one of the two processes and a transition

[⋆] Perdita.Stevens@dcs.ed.ac.uk, supported by EPSRC GR/K68547 from that process, say $E \xrightarrow{a} E'$. Eloise is required to chose a matching transition, that is, one with the same action a, from the other process, yielding a new pair of processes (E', F'). Abelard then gets to move again; because this is a *bi*simulation game he again gets a free choice of process. A player wins if the other player cannot go, and Eloise wins all infinite plays. It is not hard to see that Eloise has a (history-free[1]) winning strategy iff the processes are strongly bisimilar, and that such a strategy defines a bisimulation. Similarly, if the processes are not bisimilar, then Abelard has a history-free winning strategy. More recently, Stirling has defined and studied model-checking games[23]; it can be shown that a system satisfies a modal mu-calculus property if and only if Eloise has a (history-free) winning strategy for a certain game, with more complex winning conditions. A winning strategy in this case is closely related to a canonical tableau; again, it can be seen as a proof object.

The formulation of a verification problem as a game[2] is often not profound, but it has a number of advantages. It seems to be a natural way to capture problems involving reactive systems; in particular, it focuses on the problem rather than the system in isolation, and provides a good paradigm within which to do on-the-fly exploration. More recently, it has been realised that the winning strategy can be used as a debugging aid when the answer to the question is not what is desired. For example, understanding the failure of a liveness or fairness condition – where there is in general no "error trace" which on its own explains the problem – can be facilitated using the winning strategy[21].

2 Abstract games and abstract interpretation

The practical motivation was that we needed a theory which could be applied in the Edinburgh Concurrency Workbench [17] to solve a large number of verification problems about infinite systems. The CWB is used as a testbed for experimenting with new process algebras, relations, etc., so we wanted to minimise the amount of theoretical (and programming) work that had to be done to allow a new verification question to be applied to infinite state systems. At present we are interested only in fully-automatic verification.[3]

In [20] we described how to define, given a ("concrete") game, an associated *set game*, such that there is a correspondence between winning strategies for the two games. The paper then presented an algorithm which, when instantiated with some functions describing the concrete game, searches for a winning strategy of the set game.

Thus, if the concrete game is known to be characteristic for a question, then so is the set game, and finding a winning strategy for the set game answers the question. The algorithm generates a winning set-game strategy which can easily be interpreted as a winning concrete game strategy and so be used for debugging in the usual way. We gave some conditions under which the algorithm is guaranteed to terminate. The algorithm cannot in general, of course, be guaranteed to terminate (since it addresses undecidable problems). To go further, we need some more detail and terminology from [20].

2.1 Games terminology

For the purposes of this paper, a game is always played between two players Abelard (abbrev. $\forall$) and Eloise ($\exists$). We refer to players A and B to mean Abelard and Eloise in either order. A game G is $(Pos, I, moves, \lambda, W_\forall, W_\exists)$ where:

- Pos is a set of positions. We use $u, v, \ldots$ for positions.
- $I \subseteq Pos$ is a set of starting positions: we insist that $\lambda(i) = \lambda(j)$ for all $i, j \in I$.

[1] History-freeness is defined in 2.1
[2] possibly a very uninteresting one: the reader is invited to invent a game characterising trace equivalence
[3] However, an early approach to model-checking infinite-state processes (including those outside any known-decidable class) was [4]: Bradfield's experimental tool [2] used a combination of interaction with the user and automatic checking. It will be interesting future work to combine the approaches.

- $moves \subseteq Pos \times Pos$ defines which moves are legal. A play is in the obvious way a finite or infinite sequence of positions starting at some $p_0 \in I$ where $p_{j+1} \in moves(p_j)$ for each j. We write p_{ij} for $p_i \dots p_j$.
- $\lambda : Pos \to \{\text{Abelard}, \text{Eloise}\}$ defines who moves from each position.
- $W_\forall, W_\exists \subseteq Pos^\omega$ are disjoint sets of infinite plays, and W_A includes every infinite play p such that there exists some i such that for all $k > i$, $\lambda(p_k) = B$.

Player A *wins* a play p if either $p = p_{0n}$ and $\lambda(p_n) = B$ and $moves(p_n) = \emptyset$ (you win if your opponent can't go), or else p is infinite and in W_A. (Notice that in general some infinite play may have no winner: such a play is said to be a draw. We will forbid this for concrete games: for their abstractions it is practically inevitable, but harmless.)

A (nondeterministic) strategy S for player A is a partial function from finite plays pu with $\lambda(u) = A$ to sets of positions (singletons, for deterministic strategies), such that $S(pu) \subseteq moves(u)$ (that is, a strategy may only prescribe legal moves). A play q follows S if whenever p_{0n} is a proper finite prefix of q with $\lambda(p_n) = A$ then $p_{n+1} \in S(p_{0n})$. Thus an infinite play follows S whenever every finite prefix of it does. It will be convenient to identify a strategy with the set of plays following the strategy and to write $p \in S$ for p follows S. S is a *complete* strategy for Player A if whenever $p_{0n} \in S$ and $\lambda(p_n) = A$ then $S(p_{0n}) \neq \emptyset$. It is a *winning* strategy for A if it is complete and every $p \in S$ is either finite and extensible or is won by A. It is *non-losing* if it is complete and no $p \in S$ is won by B. It is *history-free* (or *memoryless*) if $S(pu) = S(qu)$ for any plays pu and qu with a common last position. A game is *determined* if one player has a winning strategy.

All the concrete games we need to consider are determined, with history-free winning strategies and no draws, and this is an assumption of this work.

2.2 Basic setup of abstract games

Given a game $G^{\mathcal{C}}$ (the *concrete game*) in which $W_\forall \cup W_\exists = (Pos^{\mathcal{C}})^\omega$ (no draws), the first step in defining an abstract game $G^{\mathcal{A}}$ corresponding to $G^{\mathcal{C}}$ – in other words, an abstract interpretation of $G^{\mathcal{C}}$ – is to choose a partial order $(Pos^{\mathcal{A}}, \leq)$ to represent the abstract positions. This will be related to the concrete positions (in the present work: more generality is possible, but its usefulness is not yet clear) by the existence of $h : Pos^{\mathcal{C}} \to Pos^{\mathcal{A}}$. We require that h preserves whose turn it is: $\lambda^{\mathcal{C}}(u) = \lambda^{\mathcal{A}}(hu)$ for each u. We further require that for any abstract position $U \in Pos^{\mathcal{A}}$, if $hu \leq U$ then $\lambda^{\mathcal{A}}(U) = \lambda^{\mathcal{C}}(u)$. It is convenient to consider only abstract positions U in which every u with $hu \leq U$ has the same player to move; that is, we insist that $Pos^{\mathcal{A}}$ is the disjoint union of $Pos^{\mathcal{A}}_\forall$ and $Pos^{\mathcal{A}}_\exists$, each of which becomes a complete lattice on adding a bottom element – which, however, is never a legal abstract game position. Thus if $U \leq V$ are abstract positions in $Pos^{\mathcal{A}}$ then $\lambda^{\mathcal{A}}(U) = \lambda^{\mathcal{A}}(V)$. We will write $\sim$ for the equivalence relation on $Pos^{\mathcal{C}}$ defined by $u \sim v$ iff $hu = hv$. We extend both $\sim$ and $\leq$ pointwise to sequences of positions, such as plays. In the usual way h generates Galois connections, one for each player. Given $U \subseteq Pos^{\mathcal{C}}$ with the same player to move from each position in U, $\alpha(U) = \bigsqcup_{u \in U} hu$. The assumption on U ensures that this is defined.

The canonical example (the set game of [20]) is that in which we take $Pos^{\mathcal{A}} \subset \mathcal{P}(Pos^{\mathcal{C}})$ to be the non-empty sets of concrete positions from which the same player is to move, and say that $h : u \mapsto \{u\}$. In this paper we will also be interested in certain non-trivial maps h. In particular, this will enable us to recover the decidability result of [20] in a more elegant and general form, independent of the algorithm used there.

We take $I^{\mathcal{A}} = h(I^{\mathcal{C}})$.[4]

The legal moves of the abstract game are required to satisfy

$$v \in moves^{\mathcal{C}}(u) \Rightarrow hv \in moves^{\mathcal{A}}(hu)$$

(Something weaker might still be interesting, in fact, cf the homomorphism condition of [19].) This ensures that if $p = (p_i)_{i \in I}$ is a legal play of $G^{\mathcal{C}}$ then $\overline{\alpha}(p) =_{def} (h(p_i))_{i \in I}$ is a legal play of $G^{\mathcal{A}}$.

[4] Permitting joins of elements here poses no serious problems, but complicates the presentation.

The sets of infinite plays won by each player are defined by $P = (P_i)_{i\in\omega} \in W_A^{\mathcal{A}}$ iff either for all but finitely many i, $\lambda^{\mathcal{A}}(P_i) = B$, or both

1. $\exists p = (p_i)_{i\in\omega} \in W_A^{\mathcal{C}}$ s.t. $\overline{\alpha}(p) \leq P$ (P subsumes some A-won concrete play)
2. $\forall p = (p_i)_{i\in\omega} \in W_B^{\mathcal{C}} \quad \overline{\alpha}(p) \not\leq P$ (P subsumes no B-won concrete play)

(An infinite play which subsumes no concrete play, or which subsumes concrete plays won by each player, is drawn. The first situation may arise in practice when every finite prefix of an abstract play subsumes some concrete play, but, for example, some natural number parameter of an early position has to be larger the longer the prefix. The latter can be illustrated by "sticking together" appropriate infinite plays won by each player, but is easily excluded in practice.)

2.3 Further restrictions

In order to obtain strategy transfer results, we will need the equivalence relation $\sim$ on concrete positions induced by h to be consistent with the moves relation on the concrete game, as follows:

1. (gMM) If $u' \in moves(u)$ and $u' \sim v'$ then there is v such that $v' \in moves(v)$ and $u \sim v$. (This is equivalent, in the special case considered in [20], to that paper's requirement on getMaximalMove for guaranteed termination of the algorithm.)
2. (gMP) If $u' \in moves(u)$ and $u \sim v$ then there is v' such that $v' \in moves(v)$ and $u' \sim v'$. (This is equivalent to the requirement on getMaximalPredecessor in [20].)
3. (inf) If p and q are infinite concrete plays such that for each i, $p_i \sim q_i$, so that $\overline{\alpha}(p) = \overline{\alpha}(q)$, then p and q are won by the same player.

(All of these parts are trivial in the set-game case, where $u \sim v$ iff $u = v$.) Then we define the moves relation on the positions of the abstract game by

Definition 1. *$V \in moves^{\mathcal{A}}(U)$ iff*

1. *for each $u \in Pos^{\mathcal{C}}$ such that $hu \leq U$, $moves^{\mathcal{C}}(u) \neq \emptyset$ and*
2. *for each $v \in Pos^{\mathcal{C}}$ such that $hv \leq V$, there is some w such that $v \in moves^{\mathcal{C}}(w)$ and $hw \leq U$.*

It is important to notice that there is no reason why everything subsumed by U should have a successor subsumed by V: in the set-game case, for example, a different move may be required from concrete positions in the same set, depending on the values of parameters.

Given these definitions we may observe that

Lemma 1. *1. $v \in moves^{\mathcal{C}}(u) \Rightarrow hv \in moves^{\mathcal{A}}(hu)$ (as required by the abstract game specification).*
2. p is extensible iff abstract play $\overline{\alpha}(p)$ is extensible. Player A wins concrete play p if and only if player A wins $\overline{\alpha}(p)$.
3. (Dropping) If $V \in moves^{\mathcal{A}}(U)$ and $V' \leq V$ then $V \in moves^{\mathcal{A}}(U)$.
4. (Covering) If V is minimal, say hv, and in $moves^{\mathcal{A}}(U)$, there is some minimal $U' \leq U$, say hu, such that $V \in moves^{\mathcal{A}}(U')$ and $v \in moves^{\mathcal{C}}(u)$.
5. (Lifting) If $U \leq V$ and there is some legal move from V and $U' \in moves^{\mathcal{A}}(U)$, then $U' \in moves^{\mathcal{A}}(V)$.

2.4 Strategy transfer

We now have enough to transfer strategies and show that this preserves winningness.

A strategy S is said to be *downwards closed* if whenever $p \leq q$ are both legal plays and q follows S then p follows S.

Lemma 2. *If T is any winning strategy for $G^{\mathcal{A}}$ then there is a downwards closed winning A-strategy for $G^{\mathcal{A}}$.*

Let S be a strategy for $G^{\mathcal{C}}$. Construct a strategy $\overline{\overline{\alpha}}(S)$ for $G^{\mathcal{A}}$ by

$$(\overline{\overline{\alpha}}(S))(P) = \{hu : \exists p \in S \ s.t. \ \overline{\alpha}(p) \leq P \text{ and } \ u \in S(p)\}$$

Conversely, let S be a strategy for $G^{\mathcal{A}}$. Construct a strategy $\overline{\overline{\gamma}}(S)$ for $G^{\mathcal{C}}$ by

$$(\overline{\overline{\gamma}}(S))(p_{0n}) = \{u \in moves^{\mathcal{C}}(p_n) : \exists P \in S \ s.t. \ \overline{\alpha}(p) \leq P \text{ and } hu \leq V \in S(P)\}$$

Lemma 3. *1. Considered as maps on strategies regarded as sets of plays, $\overline{\overline{\alpha}}$ and $\overline{\overline{\gamma}}$ are monotonic with respect to subset inclusion. Moreover if $\overline{\overline{\alpha}}S \subseteq T$ then $S \subseteq \overline{\overline{\gamma}}T$.*[5]

2. $\overline{\overline{\alpha}}(S)$ is downwards closed.

3. If a finite play P_{0n} follows $\overline{\overline{\alpha}}(S)$ and $h(p_n) \leq P_n$ then there is some concrete play p_{0n} ending in p_n and following S.

4. If T is downwards closed, then (the legal concrete play) p follows $\overline{\overline{\gamma}}(T)$ exactly when $\overline{\alpha}(p)$ follows T.

Theorem 1. *1. If S is a winning strategy for $G^{\mathcal{C}}$ then $\overline{\overline{\alpha}}(S)$ is a winning strategy for $G^{\mathcal{A}}$.*

2. If T is a downwards closed winning A-strategy for $G^{\mathcal{A}}$ then $\overline{\overline{\gamma}}(T)$ is a winning strategy for $G^{\mathcal{C}}$.

2.5 Relationship with the algorithm of [20]

Fundamentally the algorithm is based on the set game, that in which $h : u \mapsto \{u\}$. It works with sets of concrete positions, so it needs a way to describe those sets and manipulate them. Accordingly, to instantiate the algorithm for a particular problem requires a *notion of shape* on the positions of the concrete game, and implementations of certain functions. That is, there is a set of shapes; shapes may be parameterised on data values, clock times, etc. Each concrete position can be uniquely represented by giving its shape and values for the parameters; initially we may consider the abstract game in which h maps a position u to its shape. To make this accord with the definition, we require the notion of shape to ensure that if concrete positions u and v have the same shape (we write $u \approx v$) then $\lambda(u) = \lambda(v)$. For example, a position in a model-checking game might be the (state, subformula) pair $(\overline{a}(5).0, \langle b \rangle T)$. An appropriate notion of shape might give the shape of the position as $(\overline{a}(p_1).0, \langle b \rangle T)$, so that the position is defined by giving its shape and specifying that $p_1 = 5$. Alternatively, since the subformula in a model-checking game position determines who is to move, the whole state might be regarded as a parameter, so that the shape would be $(p_1, \langle b \rangle T)$ and the position is defined by specifying $p_1 = \overline{a}(5).0$. The notion of shape gives a coarse equivalence relation on concrete positions, which, effectively, the algorithm refines. Rather than giving specific values for the parameters, we may specify a constraint on the values and so define a set of concrete positions all having the same shape.

The equivalence relation $\approx$ itself will not in general satisfy the necessary conditions. We do require $\approx$ to satisfy condition (inf) (in fact, a stronger version of it), which is satisfied by the obvious notions of shape in the standard examples. The algorithm alternately searches and backtracks, splitting $\approx$-equivalence classes wherever necessary to satisfy the conditions (gMM) and (gMP). If it terminates, it can be seen as having constructed, *automatically*, a finite refinement $\sim$ of $\approx$ on the reachable concrete positions satisfying the conditions, and having taken advantage of the finiteness of the abstract game tree, determined by this equivalence relation, to solve the problem in that domain. Proposition 1 of [20] observed that if such an equivalence relation existed, then the algorithm would terminate: the original proof was founded on the observation that the algorithm would never split equivalence classes of such a relation.

The algorithm takes an abstract position I described as a shape and constraint, that is, a set of concrete positions, and partitions it into subsets $I_\forall$, $I_\exists$ such that each player A has a winning strategy for the game starting at I_A, if this set is non-empty. (Of course in the case that both sets are non-empty, the player whose turn it is at I has a winning strategy from I itself: this is not what is of interest.)

[5] [20] claimed parenthetically that α and γ form a Galois connection: but in fact the reverse implication does not seem to hold in general, though it does hold for history-free strategies, for example. Some adjustment to definitions may be desirable here?

3 Discussion, conclusion and further work

We have described how recent work on abstract games can be seen as abstract interpretation, generalising previous approaches to model checking by abstract interpretation. The approach however is not limited to model-checking, but applies to any verification problem which can characterised by a suitable concrete game.

A question we haven't discussed is "abstract interpretation of *what*?". We have deliberately avoided any distinguished "program", since verification *problems* do not necessarily have an obvious concept of *program*. In the case of model-checking, where the problem is to decide whether a system satisfies a formula, it is in some ways attractive to regard the system as the program, and as far as the author is aware this is what a.i. approaches to model-checking (see [12], [7], [8], [9], and many others) have done. However, this approach encounters problems when we try to extend it to model checking of logics which themselves involve potential infinite value sets, such as the first order mu-calculus of [11] or [18], or the observational mu-calculus of [3]. Here the need to distinguish a pair of values may arise from a complication in the formula, rather than from the process; it is possible, for example, for a formula to retain information about a process's previous behaviour even if the process itself does not. More seriously, it does not fit well with other verification problems such as equivalence checking in which the question is whether two systems are related in a particular way. It is possible to regard one of them as the program and the other as part of the question; but this is arbitrary and rather unnatural. We conclude that abstract interpretations are really interpretations of a problem, whose statement may or may not involve a distinguished program.

Although, as the discussion of the algorithm shows, such ideas are lurking, we have not explicitly discussed any cases in which partial answers are obtained. The main barrier is the duality inherent in the problems we consider: there is not a priori any reason to treat one player differently from the other. In practice, however, it may be that the user of a tool expects the answer to be a certain thing, and it might be possible to take advantage of this. It may be necessary to consider restrictions of the full mu calculus. It might also be hoped that further understanding of how an algorithm can move between different abstract interpretations of the same game – for example, of the connection between this and the a.i. ideas of widening and narrowing – may lead to more efficient and understandable algorithms for finding winning strategies of games.

3.1 Acknowledgements

I thank Samson Abramsky for helpful comments and pointers into the abstract interpretation literature.

References

1. R. Alur and D.L.Dill, Theory of Timed Automata, Theoretical Computer Science 126(2) pp183-235, 1994
2. J. Bradfield. A proof assistant for symbolic model checking. In Proceedings of CAV'92, LNCS 663.
3. J. Bradfield and P. Stevens, Observational mu-calculus. To appear in Proceedings of the Workshop on Fixed Points in Computer Science (FICS'98).
4. J. Bradfield and C. Stirling, Local model checking for infinite state spaces. Theoretical Computer Science, 96 pp 157-174 (1992).
5. K. Cerans, Decidability of bisimulation for parallel timer processes, In Proceedings of CAV'92, LNCS 663.
6. D. Clark, L. Errington and C. Hankin, Static Analysis of Value-Passing Process Calculi, in Theory and Formal Methods 1994.
7. E.M.Clarke, O. Grumberg, D.E.Long, Model Checking and Abstraction, ACM-TOPLAS, Vol 16, No. 3, May 1994.
8. R. Cleaveland, P. Iyer, D. Yankelevich, Optimality in Abstractions of Model Checking, LNCS 983 pp51-63, 1995.

9. R. Cleaveland and J. Reily. Testing-based abstractions for value-passing systems. Proceedings of CONCUR'94, LNCS 836, pp417-432.
10. P. Cousot and R. Cousot, Abstract Interpretation Frameworks, Logic and Computation 2(4) pp511-547, 1992.
11. M. Dam, Model Checking Mobile Processes, Information and Computation 129, 35-51, 1996.
12. D. Dams, R. Gerth, O. Grumberg, Abstract Interpretation of Reactive Systems, ACM-TOPLAS, Vol. 19, No. 2, march 1997.
13. M. Hennessy and H. Lin, Symbolic Bisimulations. Theoretical Computer Science, 138:353-389, 1995.
14. B. Jonsson and J. Parrow, Deciding bisimulation equivalences for a class of non-finite-state programs. Information and Computation, 107(2) pp 272-302, Dec 1993.
15. H. Lin, Symbolic Transition Graph with Assignment. Proceedings of CONCUR'96, LNCS 1119, pp50-65.
16. R. Milner, Communication and Concurrency. Prentice Hall 1989.
17. Moller, F. and Stevens, P. (1996). The Edinburgh Concurrency Workbench user manual. http://www.dcs.ed.ac.uk/home/cwb.
18. J. Rathke, Symbolic Techniques for Value-Passing Calculi. PhD. thesis, University of Sussex, 1997.
19. D.A. Schmidt, Trace-based Abstract Interpretation of Operational Semantics, draft from http://www.cis.ksu.edu/ schmidt/, 1998.
20. P. Stevens, Abstract games for infinite state processes. To appear in Proceedings of CONCUR'98.
21. P. Stevens and C. Stirling, Practical Model-Checking using Games. Proceedings of TACAS'98, LNCS 1384, pp 85-101
22. C. Stirling, Modal and temporal logics for processes. LNCS 1043 pp149-237. 1996.
23. Stirling, C. Local model checking games. *Lecture Notes in Computer Science*, **962**, 1-11. 1995

Appendix: proofs

Lemma 1

Proof. 1. Suppose $v \in \mathit{moves}^{\mathcal{C}}(u)$. We need to show that for each $v' \sim v$ there is $u' \sim u$ such that $v' \in \mathit{moves}^{\mathcal{C}}(u')$ – exactly (gMM).

2. If $v \in \mathit{moves}^{\mathcal{C}}(p_n)$ then (gMM) shows that $hu \in \mathit{moves}^{\mathcal{A}}(h(p_n))$, so if p is extensible then so is $\overline{\alpha}(p)$. If $\overline{\alpha}(p)$ is extensible, say by $V \in \mathit{moves}^{\mathcal{C}}(h(p_n))$, then by deinition for any v with $hv \leq V$ we have $v \in \mathit{moves}^{\mathcal{C}}(p'_n)$ for some $p'_n \sim p_n$; then by (gMP) there is some $v' \sim v$ with $v' \in \mathit{moves}^{\mathcal{C}}(p_n)$, so p is extensible.
   Suppose p and $\overline{\alpha}(p)$ are finite. Then since if one is extensible so is the other, we need only consider the case where they are infinite. If p is infinite and in $W^{\mathcal{C}}_{A}$ then $\overline{\alpha}(p)$ subsumes a play won by A. By (inf) it cannot also subsume a play won by B, so by the definition of $W^{\mathcal{A}}_{A}$ it is won by A. Conversely, suppose p is an infinite concrete play and $\overline{\alpha}(p)$ is won by A. Then $\overline{\alpha}(p)$ cannot subsume any play won by B, so since it subsumes p and the concrete game has no draws, p is won by A.
3. Straight from definition of moves: $\forall$ has a more restricted domain.
4. From gMP and definition of moves: given $U \to hv$ we know there's $u \to v$ s.t. $hu \leq U$. Then $hu \to hv$ is also legal, since $v' \sim v$ and $u \to v$ implies exists $u' \to v'$.

Straight from the definition of moves, again: $\exists$ has a wider domain.

Lemma 2

Proof. (Sketch) Let T be a winning strategy. The obvious downwards closure of T need not itself be winning. This slight problem stems from the fact that moves need not alternate in our formulation of games. In a sequence of positions throughout which it's A's turn, a winning A-strategy T might choose to remain abstract not because it can really deal with all concretions of that abstract play, but just because it knows it will have another chance to concretise in a moment. The downwards closure might concretise in a way we didn't want, preempting the intended concretisation. To deal with this, the idea is first to add segments of plays consisting of sequences of A-choices which concretise as soon as possible, and then to remove anything troublesome.

Suppose $P_i \dots P_j$ is a segment of some play in T where $\lambda(P_i) = \lambda(P_{i+1}) = \ldots = \lambda(P_{j-1}) = A$ and $\lambda(P_j) = B$. Consider all the plays formed by replacing $P_i \dots P_j$ with $Q_i \dots Q_j$ for any legal segment $Q_i \dots Q_j \leq P_i \dots P_j$, with the restriction that if B had no moves from P_j we only consider Q_j such that the same holds. Let T^+ be the strategy formed by adding in all such plays. If Q is any play in $T^+ \setminus T$, $Q \leq P$ for some play $P \in T$; and P is won by A. Now there must be some concrete play q such that $\overline{\alpha}(q) \leq Q$, because since A wins P, P must subsume some concrete play r won by A, and after P_j P and Q are the same. So $P_{j+1} = Q_{j+1} \geq h(r_j)$, and $Q_{j+1} \in \mathit{moves}^{\mathcal{A}}(Q_j)$ so by Dropping $Q_{j+1} \in \mathit{moves}^{\mathcal{A}}(h(r_j))$. So there is some r'_j with $h(r'_j) \leq Q_j$ and $r_{j+1} \in \mathit{moves}^{\mathcal{C}}(r'_j)$. So $h(r'_j) \leq Q_j$; repeat, to define q. Therefore Q is also won by A. T^+ is still complete, since anything B can move to from a Q_j could also have been reached from a P_j by Lifting. So $T+$ is still a winning strategy. Now form T' from T^+ by discarding any play which subsumes something not in T'. This doesn't destroy completeness because there are at least the minimal (concrete) sections left. Repeat (transfinitely) for all segments. The result is a downward closed winning strategy.

Lemma 3

Proof. 1. Monotonicity is clear: in each case augmenting S widens the scope of a $\exists$ quantifier. Suppose $\overline{\overline{\alpha}}(S) \subseteq T$ and take $p \in S$: we must show that $p \in \overline{\overline{\gamma}}(T)$. Suppose for a contradiction that p_{0n} follows $\overline{\overline{\gamma}}(T)$ but $p_{n+1} \notin \overline{\overline{\gamma}}(T)(p_{0n}$. Now $\overline{\alpha}(p_{0(n+1)}) \in \overline{\overline{\alpha}}(S)$, since by definition of $\overline{\overline{\alpha}}$, we always have $\overline{\alpha}(S) \subseteq \overline{\overline{\alpha}}(S)$ (but not the reverse inclusion, since the LHS includes only ground plays). So by assumption $\overline{\alpha}(p_{0(n+1)}) \in T$, and we may use this in the definition of $\overline{\overline{\gamma}}(T)$ to get that $p_{0(n+1)} \in \overline{\overline{\gamma}}(T)$, contradicting the choice of n.

If S and T are history-free strategies, then so are $\overline{\overline{\alpha}}(S)$ and $\overline{\overline{\gamma}}(T)$ and we may regard strategies as partial functions from positions to positions; then $(\overline{\overline{\alpha}}S)(u) = h(Su)$ and $(\overline{\overline{\gamma}}T)(u) = h^{-1}(Tu)$. Then $S \subseteq \overline{\overline{\gamma}}(T)$ iff for all u, $Su \subseteq h^{-1}(Tu)$, which holds iff for all u we have $h(Su) \subseteq (Tu)$, iff $(\overline{\overline{\alpha}}S) \subseteq T$.

2. Immediate from definition of $\overline{\overline{\alpha}}(S)$: it only prescribes move of the form hu, which are minimal.
3. At the last position A chose, we certainly had an underlying concrete play by definition of $\overline{\overline{\alpha}}(S)$. To get back to there, pick some minimal $h(p_n) \leq P_n$ and use Dropping and Covering. This yields a concrete play-section from the last A choice to the end (and all positions were chosen by B so within this section there's no question about whether we follow S). Problem is that the concrete position we end up with at the last A-choice may not be the one actually used in the strategy definition. But we use gMM to alter the concrete tail without changing the minimal abstract play above.
4. First suppose that $\overline{\alpha}(p)$ follows T and that p is a legal play. Just take $P = \overline{\alpha}(p)$ and $V = h(p_i)$ in the definition of $\overline{\overline{\gamma}}$ to show that $p \in \overline{\overline{\gamma}}(T)$. Next suppose $p \in \overline{\overline{\gamma}}(T)$ and consider whether $h(p_n) \in T(\overline{\alpha}(p_{0(n-1)}))$. We know there is some $PV \in T$ with $\overline{\alpha}(p_{0(n-1)}) \leq P$ and $h(p_n) \leq V \in T(P)$; that is, $\overline{\alpha}(p_{0n}) \leq PV \in T$. Since T is downwards closed, $\overline{\alpha}(p_{0n}) \in T$ which is what has to be proved.

Theorem 1

To make the induction go through we need one additional notion. Given a strategy S for $G^{\mathcal{C}}$, the $\sim$-closure $\tilde{S}$ is defined by $v \in \tilde{S}(p)$ iff there are v' v and p' with for each i, p_i p'_i, with $v' \in S(p')$. If S is a winning strategy, so is $\tilde{S}$.

Proof. (i) Suppose S is a winning A-strategy for $G^{\mathcal{C}}$. First, we must show that $T = \overline{\overline{\alpha}}(S)$ is complete, that is, prescribes at least one move at every reachable A-choice point, say from P_{0n}. But Lemma 3 shows that there is some $p = p_{0n}$ following S with $\overline{\alpha}(p_{0n}) \leq P_{0n}$. Now since S is winning p is extensible, say by $p_{0(n+1)}$, and by definition of $\overline{\overline{\alpha}}(S)$ we're done.

We must show that $\overline{\overline{\alpha}}(S)$ is a winning strategy. Let P (finite or infinite, but non-extensible) follow $\overline{\overline{\alpha}}(S)$. We construct p following $\tilde{S}$ such that $\overline{\alpha}(p) \leq P$ (except in special cases when we show A wins P anyway). Then A wins p, so A wins P provided that there isn't also a concrete play r won by B with $\overline{\alpha}(r) \leq P$. But $\overline{\overline{\alpha}}(S)$ is downwards closed, so such a play would follow $\overline{\overline{\alpha}}(S)$. Then by a technique similar to that used in the proof of 3 we see that there would have to be r' following S with $\overline{\alpha}(r') = \overline{\alpha}(r)$; so r' is won by A and we have the required contradiction.

We'll construct $p \in \tilde{S}$ from P from the beginning in chunks; each chunk except possibly the last is finite, and ends in a position chosen by A (this is the definition of a *chunk*!). Since $p \leq P$ and by the definition of $\leq$, this means that the last position in any finite chunk is p_m with $h(p_m) = P_m$. The induction hypothesis is that for some prefix P' of P consisting of $k > 0$ chunks we have constructed a concrete play $p' \in S$ such that $\overline{\alpha}(p') \leq P'$. The construction of the next chunk doesn't alter p' which is why this works for infinite plays as well as finite ones. The base of the induction $k = 1$ is Lemma 3 unless P is an infinite play with no A choice points, in which case this is the only chunk and $P \in W_A$ by definition.

For the induction step, consider the finite prefix of P consisting of the first $k+1$ chunks; write this as $P_{0m}.P''$ where P'' is the $(k+1)$th chunk. There are two cases, depending on whether P'' ends in a ground position chosen by A or not.

First suppose it does; in particular, it's finite, say $P'' = P_{(m+1)n}$. We have a concrete play p_{0m} in $\tilde{S}$ with $\overline{\alpha}(p_{0m}) \leq P_{0m}$, and must extend this underneath P''. Consider the composite $\overline{\alpha}(p_{0m})P_{(m+1)n} \leq P_{0n}$; this is legal since $h(p_m) = P_m$. Since $\overline{\overline{\alpha}}(S)$ is downward closed it follows $\overline{\overline{\alpha}}(S)$, so by definition of $\overline{\overline{\alpha}}(S)$ there is some $r_{0(n-1)}$ following S with $\overline{\alpha}(r_{m(n-1)}) \leq \overline{\alpha}(\{p_m\})P_{(m+1)(n-1)}$ (taking $P_{(m+1)(n-1)}$ empty if $m+1 = n$). Then $P_n = h(r_n)$ for some $r_n \in S(r_{0(n-1)})$.

We need not have $r_i = p_i$ where both are defined, but we do have $r_i \sim p_i$, which enables us to adjust r_{mn} to r'_{mn} with $\overline{\alpha}(r_{mn}) = \overline{\alpha}(r'_{mn})$ and $r'_m = p_m$. Now take $p_{mn} = r'_{mn}$. Then $p_{0n} \sim r_{0n}$ so since r_{0m} follows S, p_{0n} follows $\tilde{S}$.

For the other case, suppose P'' does not contain any A-chosen position (it may be finite or infinite), so that P'' is the final chunk of P. Then if P'' is finite then since it ends in a B-choice A wins as required. If P'' is infinite, then again A wins by the condition on W_A in the definition of abstract game.

(ii) Let T be a downwards closed winning A-strategy for $G^{\mathcal{A}}$. Since T is downward closed, p follows $\overline{\overline{\gamma}}(T)$ exactly when $\overline{\alpha}(p)$ follows T. The conditions on $\overline{\alpha}$ required it to preserve winners, so to show that $\overline{\overline{\gamma}}(T)$ is a winning strategy we only have to show that $\overline{\overline{\gamma}}(T)$ is complete. Now if p_{0n} follows $\overline{\overline{\gamma}}(T)$ and A should move from p_n, then since $\overline{\alpha}(p_{0n})$ follows the winning strategy T there is some abstract move, U say, such that $U \in T(\overline{\alpha}(p))$. Take $\overline{\alpha}(p_{0n})$ and this U in the definition of $\overline{\overline{\gamma}}(T)$. Pick some minimal $hu \leq U$; by downwards closure of T this is still in $T(\overline{\alpha}(p))$. Then there is $v \sim p_n$ such that $u \in moves^{\mathcal{C}}(v)$, so there is $u' \sim u$ such that $u' \in moves^{\mathcal{C}}(p_n)$; take $p_{n+1} = u'$. This is in $\overline{\overline{\gamma}}(T)(p_{0n})$ by construction.