Information Sciences 259 (2014) 36–56

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

The emergence of choice: Decision-making and strategic thinking through analogies [☆]

Alexandre Linhares [*]

EBAPE/Getulio Vargas Foundation, Praia de Botafogo 190 office 509, Rio de Janeiro 22250-900, Brazil
Center for Research on Concepts and Cognition, Indiana University, Bloomington, IN, United States

ARTICLE INFO

Article history:
Received 11 February 2010
Received in revised form 5 July 2013
Accepted 6 August 2013
Available online 14 August 2013

Keywords:
Cognitive science
Perception
Computational model
Analogy
Reasoning
Strategic thinking

ABSTRACT

Consider the chess game: When faced with a complex scenario, how does understanding arise in one's mind? How does one integrate disparate cues into a global, meaningful whole? How do players avoid the combinatorial explosion? How are abstract ideas represented? The purpose of this paper is to propose a new computational model of human chess cognition. We suggest that analogies and abstract roles are crucial to understanding a chess scenario. We present a proof-of-concept model, in the form of a computational architecture, which accounts for many crucial aspects of human play, such as (i) concentration of attention to relevant aspects, (ii) how humans may avoid the combinatorial explosion, (iii) perception of similarity at a strategic level, (iv) a state of meaningful anticipation over how a global scenario may evolve, and (v) the architecture's choice as an emergent phenomenon from the actions of subcognitive processes.

1. Introduction

Most models of decision-making and strategic thinking are based on game-theoretical ideas. Yet, the human mind works in different ways from those postulated by game theory or expected utility theory. While game theory has provided us with valuable insights into numerous scientific fields, the assumption that humans think and act "rationally" as postulated does not stand empirical enquiry [4,38]. This paper makes some observations concerning how an alternative model might be. The objective is to present a model of decision-making and strategic thought that, while admittedly preliminary and consisting of a limited proof-of-concept, may shed light on the underlying cognitive processes, and influence further work on unexplored and perhaps promising avenues of research.

Our interest here is in the bridging the gap between strategic-interaction games, experimental psychology, and work on cognitive modeling. We will be particularly interested in the game of chess, a zero-sum combinatorial game for which extensive psychological knowledge has been acquired; and the main thrust of this work is to propose that the FARG models (for Fluid Analogies Research Group) provide a serious blueprint to modeling decision-making and abstract thought in games and strategic interactions.

A number of FARG models have been devised in order to better understand perception, massive parallelism, emergent understanding from subcognitive processes, the bottom-up and top-down interplay of these processes, and the central role of analogy in cognition.

[☆] This work has been financed by Grants #470341/2009-2, #301207/2009-7, from the CNPq of the Brazilian Ministry of Science & Technology, by Grants #E-26/110.790/2009, E-26/111.846/2011 and E-26/110.540/2012 from the FAPERJ Foundation, and by the PROPESQUISA program of the Getulio Vargas Foundation.

[*] Address: EBAPE/Getulio Vargas Foundation, Praia de Botafogo 190 office 509, Rio de Janeiro 22250-900, Brazil. Tel.: +55 21 9996 2505.
E-mail address: alexandre.linhares@fgv.br

0020-0255/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.08.006

In this paper, a new computational model of chess decision-making is proposed, in the form of a computer architecture called Capyblanca. We also see how the Capyblanca project may be psychologically-plausible, gradually moving from low-level vision processing into high-level abstract reasoning.

1.1. The objective of this paper

Our interest here is on the following question: *how do chess players understand what is going on in a particular scenario*? A complete understanding of this question demands that the following scientific questions, currently without cogent explanations, be approached:

1. How can old strategies be applied to new situations?
2. When faced with a complex scenario, how does one integrate disparate cues into a global, meaningful, understanding of the situation?
3. How do humans avoid the combinatorial explosion?
4. What is an intuitive sense of a position? How are humans able to play rapidly at high-performance levels?
5. How can strategically similar positions be perceived? Which mechanisms and representation structures enable cognitive processes to perceive analogous positions that retain a shared essence?

These are the questions that guide this work. Though it is only a first step, a *proof of concept* of the ideas involved, and with obvious limitations, we make some observations concerning Capyblanca that may shed new light on these issues.

Consider question #1: *How can old strategies be applied to new situations?*

We postulate that *analogies play a key role in the understanding of a chess scenario*. Consider the positions shown on Fig. 1. Experts report that pairs (6,10) and (8,20) are 'very similar', despite the large number of superficial differences [25,26].

In positions 6 and 10, black cannot defend from the simultaneous attack of white's passed pawn and king. Note that in position 6 the white king will move across the board, to eventually threaten the E5 pawn. These movements expand the game tree, involving over 20 plies. The decision tree of position 10, on the other hand, is shallow. To the "eyes" of a traditional tree-search program, these position have *no similarity*, as only at the abstract level does their shared essence emerge. Yet, the strategic similarity to the situation in position 10 is remarkable, even with large differences in search tree depth and breadth.

Positions 8 and 20 involve exchanges. In position 8—a variant of a position used in Charness et al. [8]—, white moves rook to g8 check, black rook captures white rook, white knight captures black pawn at f7 checkmate. In position 20, white moves knight to a6 check (by knight and by queen), black king escapes to a8, white moves queen to b8 check, black rook captures white queen, white knight returns to c7 checkmate. These positions display high strategic similarity and no similarity at a surface ("appearance") level. Analogical mechanisms enable humans to perceive such abstract strategic similarities, and this enables the application of old strategies to new situations, as argued in [25,27–31].

1.2. What the paper is not

The objective of this paper is to point out a new model of chess cognition based on the psychology of chess and on FARG architectures. It is important to point out what this paper is not. The paper is focused on the innovations of Capyblanca, and not on other important topics addressed previously on the literature:

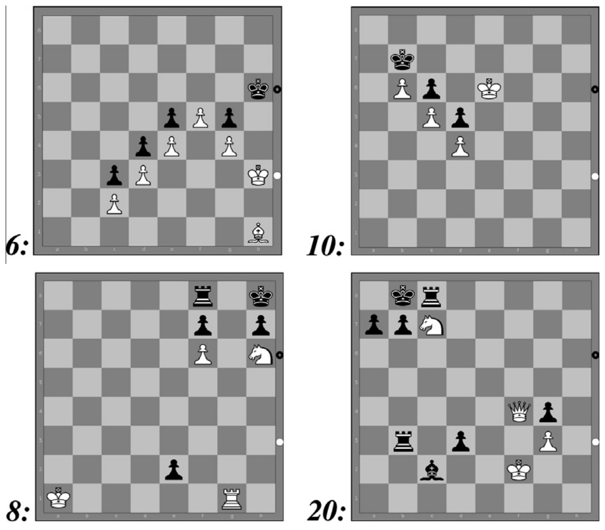


Fig. 1. White to move. Analogy enables the application of familiar strategies to new, unforeseen, situations.

- This work is not a review of the literature of computational cognitive models [14,23].
- This work is not a review of chess psychology; the reader is referred to [14–16] for chess cognitive models previous to our proposal, and to [25] for references showing that chess is a perception-based, gradual interpretation-construal process (which influenced development of Capyblanca).
- This paper does not contrast Capyblanca to traditional AI, tree-based, approaches (question #3 notwithstanding); see [25] for details on that issue. The reader is referred to [1] for a history of AI-based computer chess; and [3] provides recent advances and additional literature.
- This paper does not contrast Capyblanca to the current cognitive models of chess; see [25,27–31] arguments concerning analogies in chess and [29–30] for a critique of some well-known cognitive models of chess, including Chase and Simon's influential article [6].
- This work is not a review of FARG architectures, or the contrasts to other analogy-based cognitive models such as SME (interested readers will find detailed discussions in Hofstadter and FARG [20], Ekbia [9], Mitchell [35,36], Foundalis [10], McGraw [33], Marshall [32], Rehling [40], French [11–13]; see also Neuman and Nave [39]).

1.3. Structure of the paper

We assume that readers (or at least those who proceed from this point on) will be convinced of the use of analogy in identifying plausible strategies for any given chess position [27–30]. The remainder of the paper is concerned with the presentation of Capyblanca and the questions posed above.

Section 2 will provide the desiderata for a psychologically plausible chess model, based on the psychological literature and on previous FARG models. Section 3 details the Capyblanca project's architecture. Section 4 provides an example of the information-processing of Capyblanca and discusses psychological plausibility. Section 5 concludes by discussing how the project may help in illuminating the questions posed initially.

2. Desiderata for a psychologically plausible architecture

What are the elements of an architecture that could display understanding of a chess position? How should it process information? What does the process look like? This section presents our proposal in broad strokes.

2.1. Desiderata on Capyblanca's information-processing

The model presented here is constrained by 7 architectural principles drawn from an extensive review of the cognitive psychology of chess.

(i) *A cognitive chess architecture should not evaluate a large number of move combinations, and only sporadically evaluate move implications (i.e., search the game tree).* There is a large body of evidence that chess players do not evaluate numerous different positions, and the number seems rather small (smaller than ten) [25].

(ii) *A cognitive chess architecture should concentrate its attention (movement of eye saccades) to pieces in chess relations.* Chess players rapidly focus on the most important aspects of the board, as shown by eye-tracking experiments [25].

(iii) *A cognitive chess architecture should be able to have familiarity with real board positions, while having difficulty interpreting implausible board configurations.* Experiments in reconstruction of positions show that masters are able to reconstruct them after brief exposures (e.g., 5 s), while random positions cannot be reconstructed. Beginners have difficulty with both *game* and *random* positions. This leads to the idea of *chunking* in the encoding of chess positions [25].

(iv) *A cognitive chess architecture should manipulate a relatively small number of chunks (or other discrete structures) in short-term memory, despite having a relatively large encyclopedia of chunks stored in long-term memory.*

(v) *A cognitive chess architecture should exhibit a bottom-up parallel processing, when it is presented with a board position.*

(vi) *A cognitive chess architecture should exhibit top-down processes, triggered by acquired expectations, which should enable the program to eventually create a coherent and meaningful description of the board.*

(vii) *A cognitive chess architecture should construct fluid, rapidly-reconfigurable, multiple-leveled representations of the board (the term in the chess literature is **dynamic complexes**). These representations include pieces, relations between pieces, and, eventually, empty squares.*

The Capyblanca project postulates chess as a process of pattern-recognition with the following characteristics:

- It is a gradual perception process, creating an interpretation of the scenario at hand. Like simulated annealing, the process seems to be a "temperature-based" process, in which the space of possibilities is gradually probed in numerous areas (and multiple levels—see below). This process has been referred to as the "parallel terraced scan", and also a "gradual convergence to a context-sensitive representation" (see French and Anselme [13]).

- There are different hierarchical levels of processing. At a *low level* of pattern recognition, the process seems to be parallel and data-driven, with fast eye saccades over the most important areas of the board. On the other hand, at a high level of decision-making and inferences, the process seems to be serial and driven by hypotheses or expectations.
- Since human report protocols cannot be relied upon in order to grasp precisely the processing on a chess player's mind, we propose that the *information-processing trajectories* should be carefully probed, so that deviations of processing by the system also reflect expectations from human players of the game.

These characteristics of processing have appeared in a number of cognitive models from the FARG approach. There are numerous processes operating in parallel. Some processes are acquiring information through eye saccades across the board. Other processes may generate hypothesis and expectations, leading the system to search for additional expectations, e.g., if a piece is attacking a rook, can that piece be blocked or captured? The process works in a multiple-leveled fashion, with high-bandwidth and concrete perceptions at a low level, and gradually lower bandwidth and more abstract vision at higher levels. Here we have the first clues concerning the questions of analogy and strategic similarity, avoidance of combinatorial explosions, and integration of cues into a single interpretation, some of our guiding questions.

2.2. A brief overview of FARG architectures

FARG architectures seem particularly well-suited for such desiderata: they have simultaneous bottom-up and top-down processing, they are parallel and exhibit an emergent understanding; they proceed from high-bandwidth, concrete information, to low-bandwidth, abstract information; and their information-processing is based on a gradual construal of a multiple-leveled representation.

While we can only summarize previous FARG architectures here, let us start by distinguishing them from other notable "analogy making" architectures, such as SME or ACME. There are at least three fundamental philosophical characteristics that distinguish FARG architectures from notable analogy models: the *intention* of the research, the *nature* of the analogies, and the *processing* of representations.

2.2.1. Intention

While the term analogy-making is commonplace to FARG architectures, their intention is not to solve analogies *per se*, as *one particular cognitive capability*. Though both Copycat and Tabletop, some of the original projects of the architecture, focus on analogy-making tasks, the intention of FARG architectures is highly different from notable analogy-making theories. In fact, the intention is not the solving of analogies: it is to *model high-level cognition*, and one particular claim is that high-level cognition, deep perception, or abstract thought, emerge through analogies. The focus is on analogies *as an underlying process towards the goal of high-level cognition*, not solely as analogy-solving projects. This should be clear by the focus of the descendant projects of Copycat and Tabletop: *Letter Spirit* is a project to capture and reproduce the stylistic essence of fonts, *Phaeaco* is molded in order to solve Bongard problems, *Metacat* strives to obtain a higher level of self-awareness during its runs. More recent projects involve understanding and extrapolation of number sequences, generation of melodic expectations, discovery in Euclidean geometry, and the study of humor.

It is in this spirit that the current project is presented. Capyblanca is not a system for finding analogies in chess positions. It intends to create a "deep perception" of a chess position, and we claim that the mechanisms embodied in it are, in fact, analogy-making mechanisms.

2.2.2. The nature of analogy

In other notable theories, systems tend to assume that analogies are like a fraction: $a/b = c/d$; *a is to b as c is to d*. Systems strive to solve analogies such as the famous Rutherford analogy between the atom and the sun. Structures representing the relevant relations in both domains are presented, and the system finds an optimum matching between the structures. In FARG architectures, there is not necessarily a perfect one-to-one mapping between different structures. In fact, the system does not even have previously-carved objects and relations to start with.

Consider the following analogy: "Wikipedia is like a true democracy"—what is meant by the word "Wikipedia"? The meaning is not that of an encyclopedia; as nobody would understand the parallel analogy: "Encyclopedia Britannica is like a true democracy". It is also not meant as a website *per se;* the true intended meaning is between the *process* involved in Wikipedia sharing some similarities with the democratic process. FARG systems are built to find analogies, but the analogies do not necessarily have a one-to-one mapping. Of course one may try, *a posteriori*, to force a one-to-one mapping onto an analogy, by selecting objects and relations from each domain (as artificial as that may be; see [5,10,11,19,22]). But in FARG architectures there are no high-level objects or relations given *a priori*, which lead us to our next distinction.

2.2.3. The "perception module assumption"

As mentioned, other analogy-making systems generally receive as input a set of objects and relations construed *a priori*. In the atom-sun analogy, **atom** is an object and **sun** is an object. A set of relevant relations is also given in detail. This presupposes that the relevant information has been filtered, or rather, *can be filtered*, through perhaps a perception module. This postulated module would throw away all connotations associated with sun (enormous, massive, hot, shiny, sky, round, yellow, seems to rotate along the Earth during the day, disappears during the night, sunrise, sunset, looks smaller than a mountain, fire, star, gravity, etc.), and separate only those that are relevant. This assumption has been seriously questioned and seems duly unfeasible [5,11,12,20,24]. Unfortunately, these arguments seem to have been ignored [26]. In FARG systems there is no perception module or pre-constructed objects beyond mere atomic entities, each of which brings forth a large set of connotations and *it is therefore up to the system to filter the relevant ones from the irrelevant ones* in each particular context. A crucial part of analogy is bringing focus to some aspects of an object, relation, or situation.

FARG architectures are not hybrid architectures, but an attempt towards a synthesis of a human's cognition. They are parallel at one level, and serial at the other. They are bottom-up, data-driven, and also top-down, expectation-driven. They present a gradual convergence in which subsymbolic processes compete and cooperate to create fluid hierarchical representations (that can be rapidly reparsed).

In this family of architectures, there is a high degree of parallelism, some of which is bottom-up and data-driven. Registration of bottom-up data leads to expectations and hypotheses (i.e., a bishop primes a player to hypothesize about the contents of its diagonals). Hence, there is a number of simultaneous top-down, hypothesis-driven, processes attempting to gather more meaningful information concerning the scenario at hand, and to constrain the plausible courses of action. A result of this is that the overall process is emergent, and causality can hardly be traced to any single individual process. Another consequence is in the improved robustness of the system: if, at any point, a process has not gathered a meaningful piece of information, which could radically alter the hypotheses and expectations (and therefore all subsequent processing), the architecture may still be able to suddenly obtain such piece of information and trigger the necessary chain reaction. Erroneous, irrelevant "counterfactual" pathways are always being explored by some processes, but generally self-correct rapidly.

In the next section, the architecture of Capyblanca is presented.

3. The Capyblanca architecture

FARG architectures have been applied to many different domains, and this literature will not be reviewed in detail here, except for the concept of temperature, which has a different implementation in Capyblanca, and needs to be discussed (and justified) more closely.

3.1. Architectural elements

The architecture consists of the following five elements:

3.1.1. Pressing urges and codelets

The computational processes constructing the representations on short-term memory are subcognitive processes named *codelets*. The system perceives a great number of subtle pressures that immediately invoke subcognitive urges to handle them. These urges will eventually be executed as codelets. Some of these processes may look for particular objects, some may look for particular relations between objects and create bonds between them, some may group objects into chunks, or associate descriptions to objects, etc. The collective computation of these impulsive processes, at any given time, stands for the working memory of the model. These processes are involuntary, as there is no conscious decision required for their triggering. (As George Lakoff puts it, if one asks you "not to think of an elephant", it is too late, you already have done so, in an involuntary way.) They are also automatic, as they know how to do their job without asking for help. They are fast, with only a few operations carried out. They accomplish direct connections between their micro-perceptions and micro-actions. Processing is also granular and fragmented—as opposed to a linearly structured sequence of operations that cannot be interrupted [11,20,35]. Finally, they are functional, associated with a subpattern, and operate on a subsymbolic level (but not restricted to the manipulation of internal numerical parameters as opposed to most connectionist systems).

For example, codelets in Capyblanca can be of the following (idealized) form:

3.1.2. Coderack: a list of parallel priorities

Each codelet executes a local, incremental change to the emerging representation, but the philosophy of the system is that all the active urges are perceived simultaneously, in parallel. So there is, at any given point in time, a list of subcognitive urges ready to execute, fighting for the attention of the system and waiting probabilistically to fire as a codelet. This list of parallel priorities is called the coderack in [11,20,35]. The name is due to an analogy with a coat rack in which any coat may be taken out of from any point. The coderack is a task manager in which urges are selected stochastically to execute as codelets.

3.1.3. *A workspace that interacts with external memory*

This is the working short-term memory of the model. The workspace is where the representations are constructed, with pressing urges waiting for attention and their corresponding processes swarming over the representation, independently perceiving and creating many types of subpatterns. Common examples of such subpatterns are relations between objects, awareness of abstract roles played by objects, and so on. In the next section we will study the emergence of choice through the actions occurring in the workspace.

3.1.4. *A semantic associative network undergoing constant flux*

The long-term memory of the system is embedded into a network of nodes representing concepts with links between nodes associating related concepts. This network is a crucial part for the formation of a chain reaction of conceptual activation: any specific concept, when activated, propagates activation to its related concepts, which will in turn launch top-down expectation-driven urges to look for additional information relevant to each active concept [25]. For example, if a bishop is found at a certain position, this perception will trigger associated concepts (and codelets) to look at the diagonals, and to construct a chess relationship structure if a piece is found. This mode of computation not only enforces a context-sensitive search but also is the basis of a chain reaction of activation spreading – hence the term 'active symbols' [18,19]. This network is called the slipnet. One of the most original features of the slipnet is the ability to "slip one concept into another", in which distances between concepts change through time (for implementation details see [35]).

3.1.5. *Temperature*

Since, at any point in time, a process can propose the creation or destruction of any structure, how should such decisions be made? It is counterproductive to destruct elements that are relevant to the situation. On the other hand, the system behaves as a Brownian motion if proposals are randomly accepted. Because a proposal's quality can be measured, one may decide in the extreme to accept all "improving" proposals, leading to a hillclimbing process; or, to wonder through the search space for more promising areas to be explored. Temperature is a global variable that acts on the decision for a proposal's acceptance. The idea is to, initially, find promising areas to explore, and, as good proposals are found, temperature drops—leading to a higher probability of subsequent improvements to be made. Higher temperatures throw the system onto a breadth-first search, lower temperatures throw the system onto a depth-first search. Hofstadter poses the analogy of finding a good book in a bookstore: one usually browses around shelves, then stops at some interesting shelf, then browses some books, then selects one to browse, and if it is not of interest, puts it back on the bookshelf. No sane person would attempt to either start by reading all book titles (breadth-first search), nor start by blindly reading the first book one comes across, regardless of subject (depth-first search). As one gathers more information, this information is fed back into temperature, and a lower temperature brings a higher propensity to stay in that region of the search space; a higher temperature brings a higher propensity to move elsewhere. This process is gradual over the course of a run, and has been termed as a *parallel terraced scan*.

3.2. *Redefining the concept of temperature*

Perhaps the concept of temperature can be improved upon. Temperature stands in these models as a global feedback mechanism—all processes have direct access to the global variable T. This does not seem to be either biologically or psychologically plausible. It does not seem to be biologically plausible because the brain is massively distributed and parallel. It is questionable to suppose that a single variable is being accessed at all times by a large number of independent processes. Moreover, even if the process generated by such temperature-management techniques is much more psychologically plausible than either breadth-first search or depth-first search, we propose that the concept can be improved upon. Consider that, for each created structure, we can measure its *relevance*. Relevance is a multiple of three variables: (i) the context, or *goal* under pursuit (spawned by a top-down codelet), (ii) the ability to *imagine* the structure, and (iii) *evidence*. Each of these in isolation should seem obvious, but their combination may be a contribution to the FARG architecture literature in itself. Let us look at each of them separately.

First, relevance should be a function of the goal under pursuit. In the case of chess—though the ultimate goal is clear and unchanging—, the *current goal* is always shifting: one wants to capture *that* pawn, or one wants to block a particular file, etc. Each goal is a top-down pressure on the system; an active symbol attempting to find a set of structures which will satisfy it. There are constantly multiple goals competing for attention during a run. There is a lot of wasted effort, as many goals will trigger processes which will fail to find a proper structure. If a proposed structure leads to a higher chance of a current goal being achieved, relevance *for that structure* should increase.

Next is the ability to *imagine* a structure. The reasoning behind this ability is subtle and may sound childish at first. Our minds make it possible to imagine a "green giraffe elephant"—perhaps a dyed morph of both animals resembling some of Salvador Dali's paintings. But without a goal and specific evidence, the relevance of the proposed object cannot be understood in anyone's mind. This is but one of the crucial roles that analogy plays in cognition: given evidence (e.g., a tall politician of the American Republican Party who is concerned with environmental issues is on focus) and a goal (mocking such a person), the "green giraffe elephant" becomes a real psychological entity, and one may immediately understand the reference. Consider also the opening position of the game of chess. It is *possible* to imagine the white queen "jumping over the pawn" to land at square a4, from where it will be able to "jump again" to checkmate. But the rules of chess forbid such wishful thinking. So the ability to imagine this scenario should be valued at zero (bringing *relevance* to zero). Now, consider a bishop blocked by same-color pawns. As a pawn moves, the ability to imagine the bishop moving immediately increases from zero. This will trigger top-down processes to probe the possibilities therein. Imagination is, of course, the basis of thought experiments and counterfactuals.

Finally, relevance is a function of *evidence*. Without the tall congressman being on focus (visually or verbally), one cannot understand the term the first time it is heard. In the game of chess, the pieces on the board, and the configuration of squares that are empty, blocked, occupied by the opponent or protected by the opponent, combine to form evidence for imagined moves. Without evidence, imagination is irrelevant. And nothing is relevant without a context (provided by goals).

How are these computed? While goals are brought by top-down processes, the ability to imagine and the current evidence are brought by the contents of external memory (the chess position) or short-term memory (the current representation of the position). The values of i) *goals*, ii) *evidence* and iii) *imagination* range from $[0,\ldots,1]$ and relevance is computed by multiplying these three.

Why do we think that this distilling of the concept of temperature is needed, and indeed desirable? For both biological and psychological reasons. This proposal is massively distributed, as there is no central variable underlying all decisions—it therefore suggests itself as a more biologically plausible alternative to a global, central, *temperature* variable. And we believe it is more psychologically plausible. Consider, for example, the Copycat project. During an execution of the architecture, there are many moments in which the system is stuck at a local minima, and after a while with no improvement, temperature rises. When that happens, *every single structure created so far has a higher probability of being destroyed*. So there are scenarios in which the system goes through a particular trajectory, gets blocked, destroys everything in short-term memory, only to reconstruct a large part of what was just destroyed. Sometimes this is desirable, but for most of the time it is not (one motivation of the Metacat project was to avoid this, by realizing that it was finding itself stuck in the "same" situation). When we hear a phrase with a new, unrecognized word (e.g., "I think this is just schadenfreude in John's part"), we understand the entire phrase, while remaining confused only about a single strange part of it; there is no need to re-parse or re-interpret the entire phrase. When we analyze a chess position, our focus tends to shift between areas and relations, and we may have a clear understanding of most areas while being confused about the situation in some area. In most cases, it is unnecessary to destruct most structures. The *binding of relevance to each structure* lets us destroy (probabilistically) only those that seem either out of context (goal) or without evidence (note that if it is not imaginable, it would not be suggested in the first place).

Finally, our proposal is equivalent to temperature in an important sense: temperature can be thought of as *tolerance towards irrelevance*. A high temperature lets highly irrelevant structures in, while a low temperature demands increasing relevance. The large scale behavior of the parallel terraced scan is preserved.

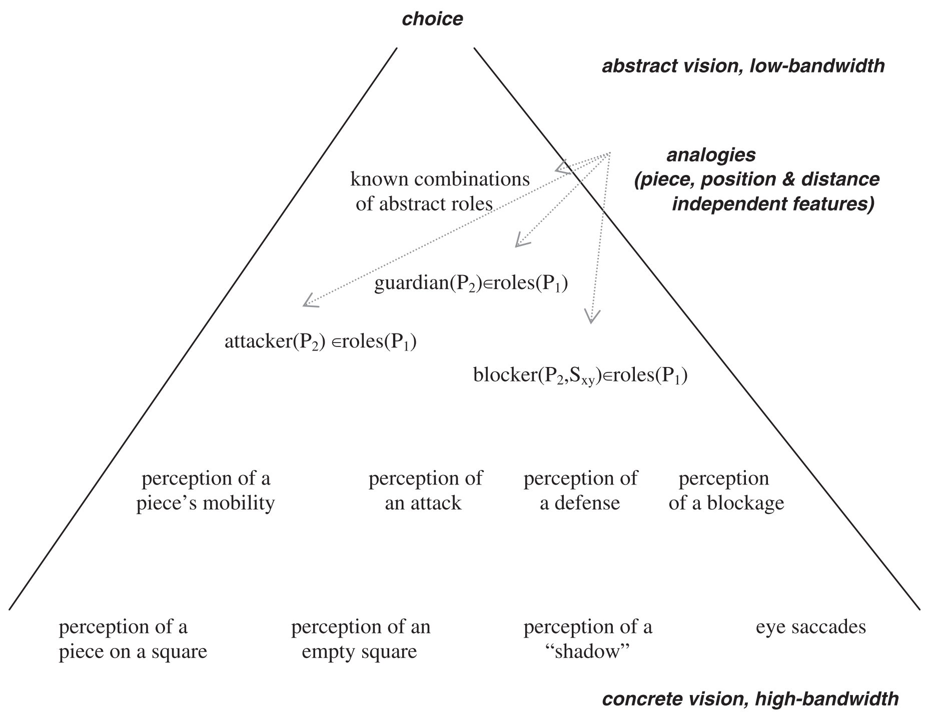

choice
abstract vision, low-bandwidth
analogies
(piece, position & distance independent features)
known combinations of abstract roles
guardian(P2)∈roles(P1)
attacker(P2) ∈roles(P1)
blocker(P2,Sxy)∈roles(P1)
perception of a piece's mobility
perception of an attack
perception of a defense
perception of a blockage
perception of a piece on a square
perception of an empty square
perception of a "shadow"
eye saccades
concrete vision, high-bandwidth

Fig. 2. The emergence of choice: as time progresses, humans move (in a non-monotonic way) from higher-bandwidth to lower bandwidth processes; from concrete observations to abstract observations. Note that there are multiple levels of perceptions involved, and interpretations will be construed by, and bind, these perceptions in all levels. From the particular pieces to the (important or irrelevant) roles these pieces are playing, a final representation should encompass information from these multiple levels. Note finally that information may flow through all of these processes, vertically and horizontally: a glance at a "shadow" (see below) can lead to the perception of a particular piece; perception of a particular piece can lead to the perception of an attack; and perception of an attack can lead to the top-down, hypothesis-driven, search for a piece capable of blocking such attack.

```
If (Bishop is found at square [x,y]) then
  . launch top-down codelets to check all squares diagonal to [x,y]
```

```
If (Piece $P_1$ is found at a Bishop's diagonal) then
 if color($P_1$)=color(Bishop) then
  . launch codelet to create Abstract_Role blocked_by($P_1$)$\in$roles(Bishop)
  . launch codelet to create Abstract_Role defended_by($P_1$)$\in$roles(Bishop)
 else
  . launch codelet to create Abstract_Role attacked_by($P_1$) $\in$roles(Bishop)
```

Fig. 3. Examples of idealized codelets operating on the levels of pieces, relations and abstract roles.

All of these architectural elements operate on many different levels of organization, some more 'concrete', tied to the specifics of a position (i.e., *percepts*), and others more 'vague', tied to abstract vision and pure reasoning. This is our next subject.

3.3. Multiple-leveled analysis

Capyblanca employs many different levels of analysis (Fig. 2). It does not start with a representation of pieces and their positions and moves onto explore the search tree. Instead, the very board position is given as "external memory", and the piece positions must be found through simulated eye saccades. The focus of the architecture is on a gradual convergence towards a representation with multiple hierarchical levels. At start, short-term memory does not contain any structure—not even the pieces. The system starts with the intention to scan the board in simulated eye saccades (given by a codelet per board square: 64 codelets only). At this stage there are no expectations, no top-down processes, and it is more like a low-level visual process than a reasoning process. Gradually, as more codelets are triggered, the processing becomes less visual and closer to a reasoning or inference mode (see Fig. 3)

As pointed out, the system operates at multiple hierarchical levels, notably:

- *Squares*—this is the lowest level of information processed by the system. Capyblanca is able to 'saccade eyes' to empty squares, to recognize that pieces may be present in its' "peripheral vision" (perceiving a "shadow", which increases probabilities of codelets in that direction), and to register that a particular piece has been found at a particular square;
- *Pieces*—After a piece is found, codelets may trigger additional codelets to search for the available trajectories of that on the following moves; other codelets will attempt to gather an estimate of the initial "spheres of influence" of the piece, leading to another level of mobilities;
- *Piece mobilities*—This level of information processing is mainly concerned with each piece's possible influence across the board. An initial map of movement is developed from the starting piece square. This map may be modified in due course by other codelets gathering additional information;
- *Trajectories*—If a piece A is found (randomly at first) to have a potential trajectory towards a piece B, then a codelet may create a data structure to encompass the trajectory between both pieces. Trajectories eventually created will lead to the creation of chess relations;
- *Chess relations*—If attack or defense relations are found, the system registers them in a proper structure, beyond the mere trajectory involved (note that multiple-steps attacks and defenses are registered, just as immediate, single step, relations). These relations will be measured by an *intensity function*, which takes the following form:

$$I = 10^{-n} v^{-1}(p_1) v(p_2) d^{-1}(p_1, p_2),$$

where p_1 represents the origin piece, p_2 the destination piece of a chess relation, $v(p_i)$ is the current system value of piece i, and v^{-1} its inverse function, d^{-1} is the inverse function of the minimum distance, in number of squares, that p_1 must travel to reach p_2. Intensity might be lowered by a codelet after this initial estimate, by increasing $n \to n + 1$ (or $n \to n - 1$, if the converse is true, under the proper circumstances). This will be illustrated in the next section (see Figs. 8 and 9). Intensities are established by (i) the value of the pieces involved in a relationship; (ii) the distance between pieces; and (iii) the mobility (or lack thereof) a piece. For example, if a low-value pawn attacks a high-value queen in 1 move, this role of attacker receives a greater priority over the relation of a high-value queen attacking over 3 moves a low value pawn. If an attack can be intercepted, the relation also loses intensity (by other codelets manipulating n—note that any of the functions v and d can be changed by codelets at any time). The idea behind manipulating n is that the same relation can be more important, or less important, as the system discovers *other* relations, and hence has the ability to manipulate intensities in a context-sensitive form. Intensity is used here as a proxy to relevance.

```
If (Attacked_by(P1) ∈roles(King) AND Attacked_by(P2) ∈roles(King)) then
  . launch top-down codelet to check whether King can capture P1
  . launch top-down codelet to check whether King can capture P2
  . launch top-down codelet to check whether King can escape to a safe square
```

Fig. 4. A codelet with an abstract role known by all chess players: under a double-check, the only solution is to escape with the king. (*Note* that in this particular idealized codelet "attack" is an immediate, next-ply, attack, and not the possibility of attack in a higher number of plies).

- *Abstract roles*—Chess relations with the highest intensities will form a set of *abstract roles*: the current interpretation of the system about what is important in a chess position, and which pieces are involved. An example of information processing at the 'abstract role' level might be: if there is a double check, the only possible response is to escape with the king. This inference might thus trigger codelets, at the piece mobility level, to check whether the king has an escape route (Fig. 4).

Reasoning *at the abstract role level*, we propose, is what enables chess players to intuitively understand a position and to perceive abstract strategic similarity to positions that are different in all superficial aspects [27–31]. Abstract roles are independent of the squares, the piece-types, the mobilities and spheres of influence, and the distances involved in the trajectories underlying chess relations. Because of this independence, abstract roles are applicable to a large set of superficially different, but strategically similar, situations. Analogies in chess are enabled by combinations of abstract roles. Also, by having high intensity attached to a codelet such as that of Fig. 4, the system is able to focus on the most important relations, and hence avoid the combinatorial explosion. We propose that most of the skill grandmasters acquire over the decades are inferences of the form provided above (with two caveats, however: first, they are mostly unconscious. Moreover, they are hardly verbalizable: because relations such as "attack" can be stored over trajectories with many movements, it is a matter of context whether or not a piece is placing pressure on another piece or in a particular area).

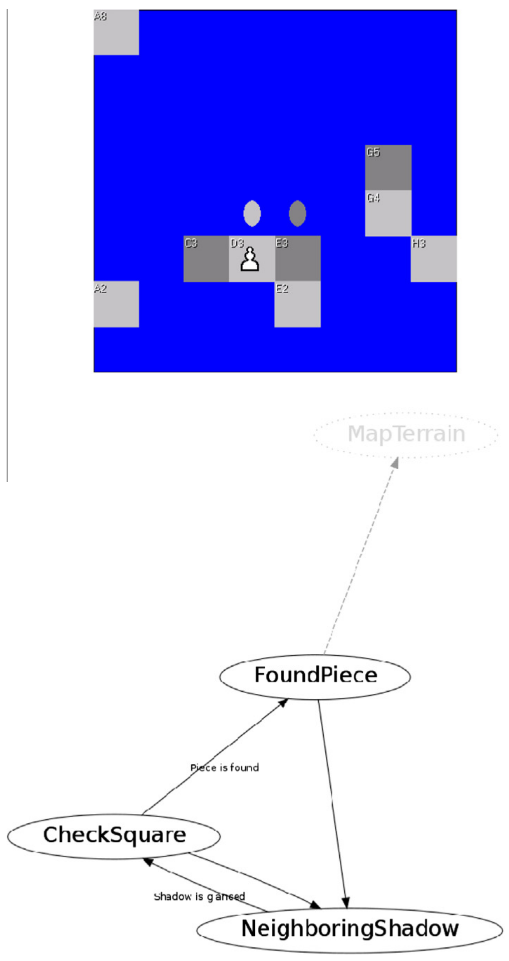


Fig. 5. State 11 of an execution of Capyblanca. The graphs following the chessboards display how the activity of the system's codelets gradually migrates from low-level feature perception to high-level abstract perception and reasoning. At this stage, the system is on a low-level vision phase.

4. The emergence of choice: an example of Capyblanca's information processing

In this section we describe an actual execution of Capyblanca in problem 6, a variation of Wilkins' [42] position. This is a much harder position, in fact, than that used by Wilkins, with increased search depth and breadth. How could a computer understand the position and escape from searching the tree in order to find white's advantage? Figs. 5–11 provide us with a careful look into the intertwined perception and interpretation processes carried out by the model.

How do experts rapidly perceive what is relevant in a position? How can they immediately zoom in the important areas of the board? In Capyblanca, the mechanisms that enable such behavior are not explicitly pre-programmed, but emerge from the interactions of the micro-perceptions of codelets. At start, the idea is to model the process akin to the first seconds of perceiving a position: the system has only 64 codelets to eventually 'saccade eyes' to each board square (codelet `CheckSquare`). Since codelets are drawn at random from the coderack, one might expect that the system would randomly scan the board. That is not the case: the board is usually scanned in a way that concentrates priority to the "most important" areas. If a piece is found by a codelet `FoundPiece`, that codelet will increase the priority given to `CheckSquare` codelets looking at the squares that the piece can move to in subsequent moments of the game. To model an ability of (a rather myopic) peripheral vision, if there exists a piece in one of the (at most 8) squares surrounding a saccaded square, then a 'shadow' is perceived, through a `NeighboringShadow` codelet (i.e., a representation that "something" is in there, but the system is unable at that point to state exactly what). In Fig. 5, for example, after codelet 11, we can see some random squares which have been saccaded to, and also, a first piece has been found through a `FoundPiece` codelet, which triggers the `MapTerrain` codelet, an event which increases the urgencies to look at *that piece's potential trajectories*. Two shadows are also found, and now the urgency to 'look at those squares' is increased. As with humans, unexpected perceptions in peripheral vision bring an urge to focus on them, and this is reflected in the system by an increased urgency to look at *that* square.

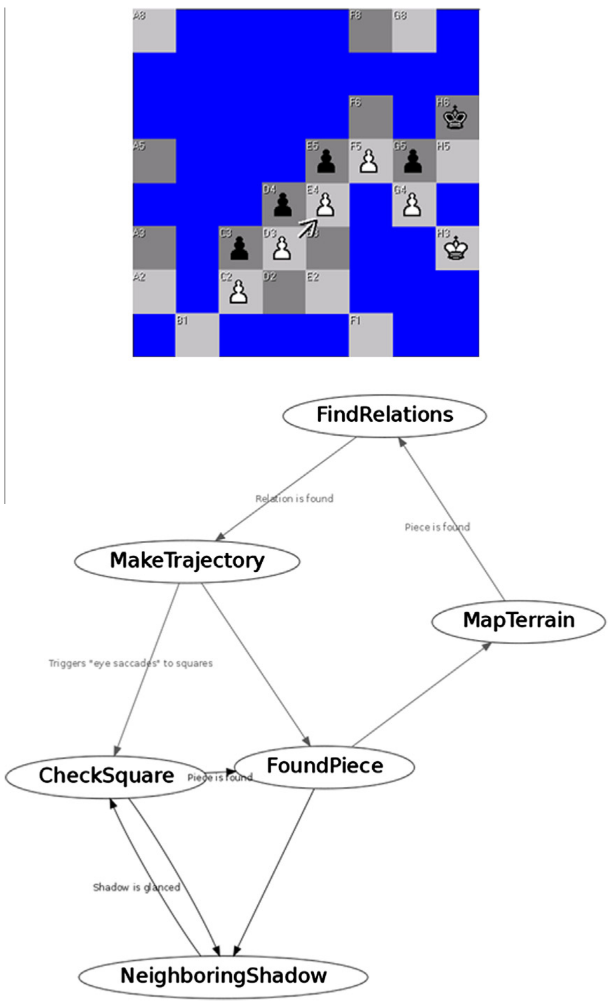


Fig. 6. State 55. There is fast convergence to the most important areas of the board: At state 55, all pieces but the bishop have been registered (as has a first defense relation through a `FindRelations` codelet).

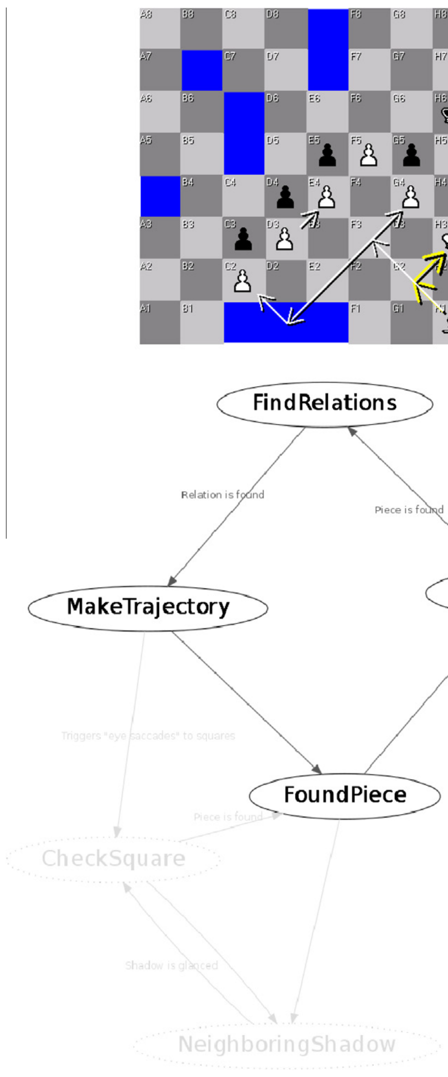


Fig. 7. State 160.

Attention rapidly converges to the important areas: at state 55 (Fig. 6), Capyblanca has found 91% of the pieces while scanning less than 38% of the board's squares. The system has also perceived a first chess relation, even if a large empty region of the board—given by the odd-colored squares—remains still unattended. At this stage, the system is processing information at five different levels: squares, shadows, pieces, piece mobilities, and immediate chess relations. Each type of information may trigger subsequent codelets and significantly affect the upcoming processing. This shows how attention is quickly thrown into the 'relevant' areas of the board. Herein lies an interesting first point concerning the system: a key mechanism in low-level perception (pieces on squares) starts from a random order of 'eye saccading', to non-random attention shifts towards the most important areas of the board (as has been documented by [8]). It is an emergent effect, a consequence of subtle shifts in urgency given the system's prior acquisition of information and launch of top-down codelets from higher hierarchical levels. It is not pre-programmed *a priori* and, at each run, the process will be different—there is no efficient algorithm or data structure to rapidly register the pieces. It is actually possible, in principle, that Capyblanca might find the pieces only after saccading to all remaining empty squares. This theoretical possibility, however, does not arise in simulations, and the large-scale, statistically emergent, effect of attention placing is robust. In this run, merely 24 out of the 64 squares have been 'saccaded to', but these tend to be the most important ones.

At state 160, note that the Bishop's "moves" are not moves in the tree-searching sense (with plies followed by the opponent). Instead, they are intended to model subcognitive pressures: the first fleeting seconds one perceives this chessboard and the Bishop's constrained area. After duly recognizing the pieces, but before fully scanning the board, Capyblanca has found that the bishop, the most powerful piece in the position, is "trying to get out", but that its mobility is restricted. Additional information displayed concerns the bishop's trajectories leading to the king and to the "unprotected" (at least at this stage) pawns. But there are new types of representational structure active at this point:

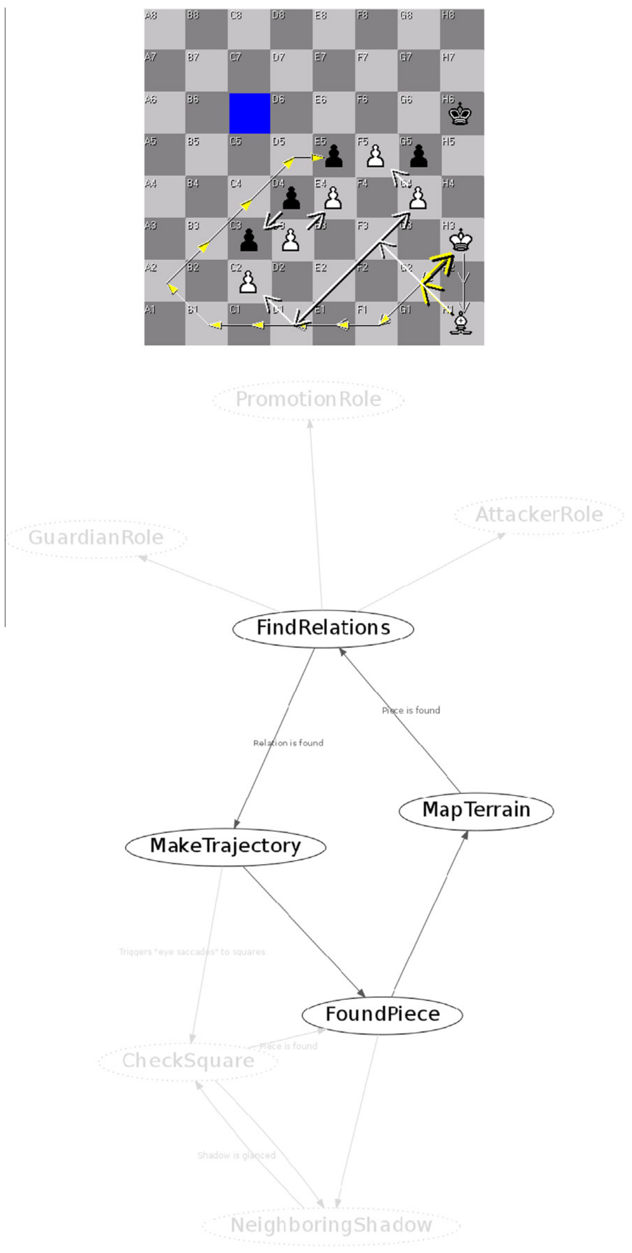


Fig. 8. State 220.

piece mobility and trajectories. After a piece has been recognized and (some of) its potential trajectories discovered, a 'mobility matrix' is constructed to account for that piece's 'spheres of influence' across the board. This information will enable many upcoming events, and, more importantly, will restrain the possibilities for consideration (not only for that piece, but also, globally across the board). Technically, the `MapTerrain`, `FindRelations`, and `MakeTrajectory` codelets register and search for the possible moves of a piece, the relations it may have with other pieces, and the trajectories involved in these relations. One should note the decreasing activity of the low-level `CheckSquare` and `NeighboringShadow` codelets as the board perceiving phase approaches its end.

After codelet 220, the white king is found, through random chance, to be able to attack the pawn at e5, and proper trajectories and roles are triggered. Note that the pawn at d4 might also have had been a possible candidate, for, at that stage, Capyblanca had yet to find that square d4 was protected by black's pieces. The system is yet to find that this new trajectory is not sustainable. In this point, all different hierarchical levels shown in Fig. 2. are simultaneously being processed by the architecture. There are `CheckSquare` codelets still 'waiting' to saccade to square c6, `MapTerrain` codelets 'waiting' to create the 'spheres of influence' matrix of a certain piece, `FindRelation` codelets waiting to create a certain type of chess relation, `MakeTrajectory` codelets 'waiting' to register a trajectory for a certain chess relation, among others less active. The low-level perception phase is over and the system is about to enter a phase of "high-level abstract reasoning".

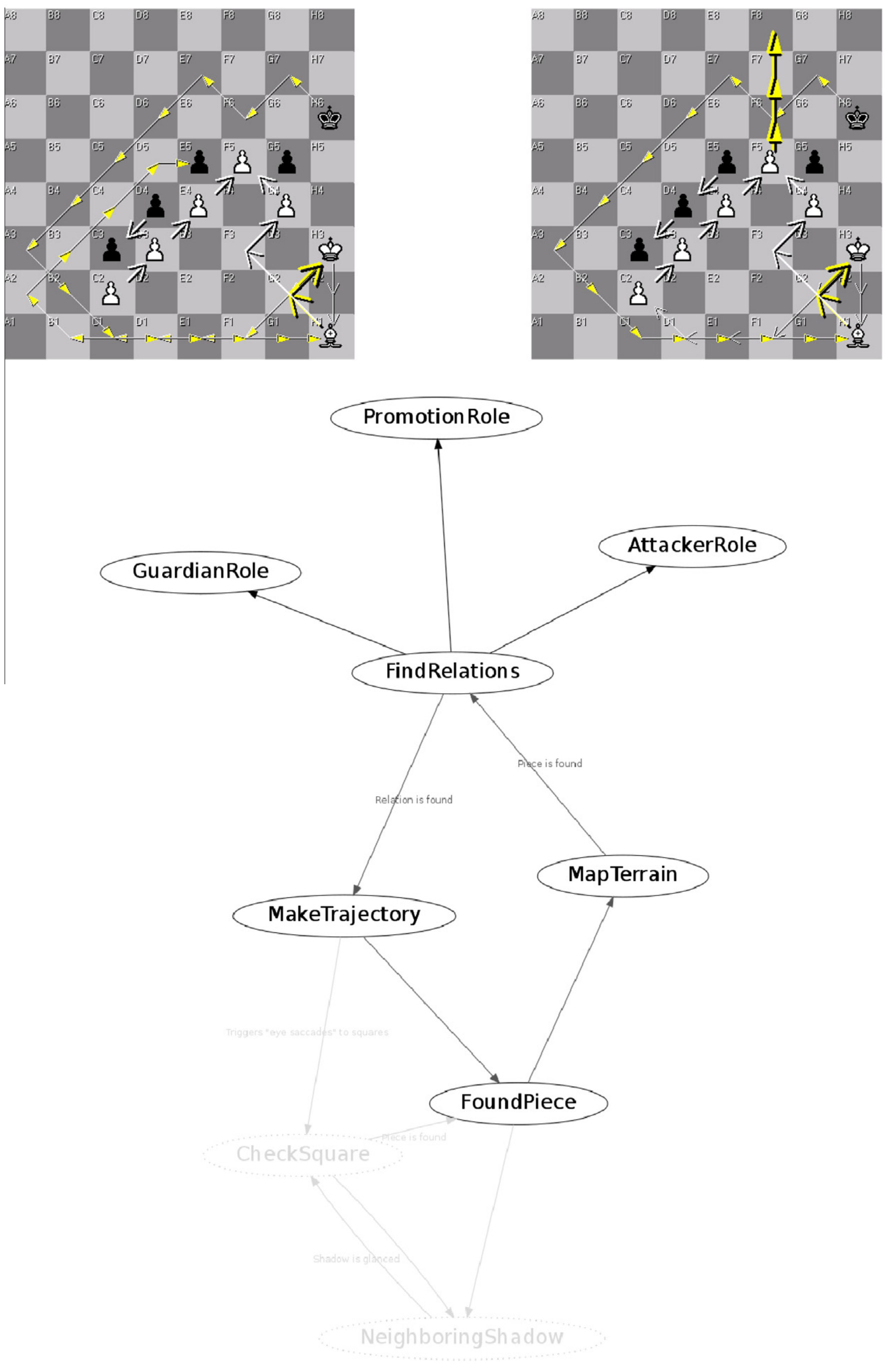


Fig. 9. States 320 and 384.

After codelet 320 (Fig. 9), the black king, again through random chance, is found to be able to attack the white bishop. The new type of representation structure active at this point is that of ‘abstract roles’. For example, the bishop is seen to play the role of ‘guardian’ of one pawn and of the king’s square. The current intensity of these roles is high, as displayed by the thickness of the arrows. Capyblanca is attempting to find escape routes (and uses, or roles) for the bishop (readers may please take the anthropomorphic terms in this narrative with the appropriate grain of salt). Note the different boldness in the board’s arrows, indicating what Capyblanca perceives to be intensity of the threats and defenses. This configuration remains relatively stable (under slight changes) until codelet 384, when the discovery that white’s passed pawn can possibly promote is made through a `PromotionRole` codelet.

After codelet 384, the system perceives the potential passed pawn promotion, a “local event” which triggers a fast, global, chain reaction of repercussions across the entire board. The system immediately places a high-urgency top-down codelet to check whether the promotion can be blocked by any of black’s pieces. These repercussions follow rapidly, so we must “slow down” this narrative to intervals of 5 (or less) codelets in the following figures. Note also that the white king’s attack on e5 is found to be easily blocked by the black king, thus the system has destroyed that role (and associated trajectory).

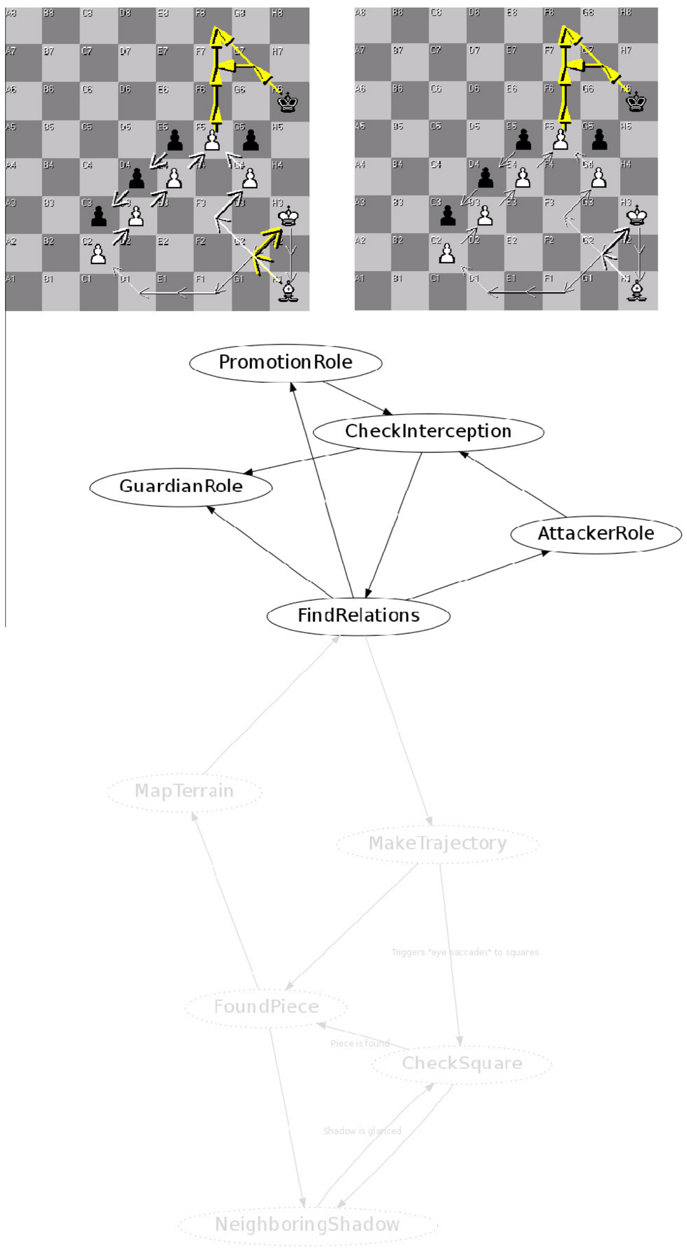


Fig. 10. States 388 and 390.

After codelet 388 (Fig. 10), the low-level perception phase is complete, and the registration, through a `PromotionRole` codelet, of the major threat (in terms of intensity) emanating from the passed pawn triggers a search for black's potential interceptions to the promotion (`CheckInterception` codelets). Those triggered codelets have high urgency and rapidly discover that the black king is able to intercept the trajectory and defend from the promotion. Again, trajectories and roles are created. But, in this case, the processes also perceive that the black king is *the only piece* able to restrain the pawn from promoting, and hence codelets restrain the black king's mobility to squares in which it would be unable to respond to the threat: the black king's spheres of influence are reduced, therefore, to a subset of the squares in the rectangle defined by [c5, h8].

At this point the system is focused only on high-level abstract processing: roles that pieces play, and how those roles can change if perceived in new ways. Since the black king, and by consequence the black side, can no longer attack the white side's unprotected pawn, two states later, after codelet 390, the roles of 'guardians' of the white pieces lose intensity. In the absence of need, the 'guardian' roles in the white side lose intensity (by increasing *n*, as discussed previously), through a `GuardianConsiderations` codelet—see the commented code in Appendix A. This triggers a search for other relevant roles those pieces might have. Note that this inference concerning intensity of the guardian relations is achieved by information processing at the abstract role level (the black king is no longer a threat). The reader may notice that processing

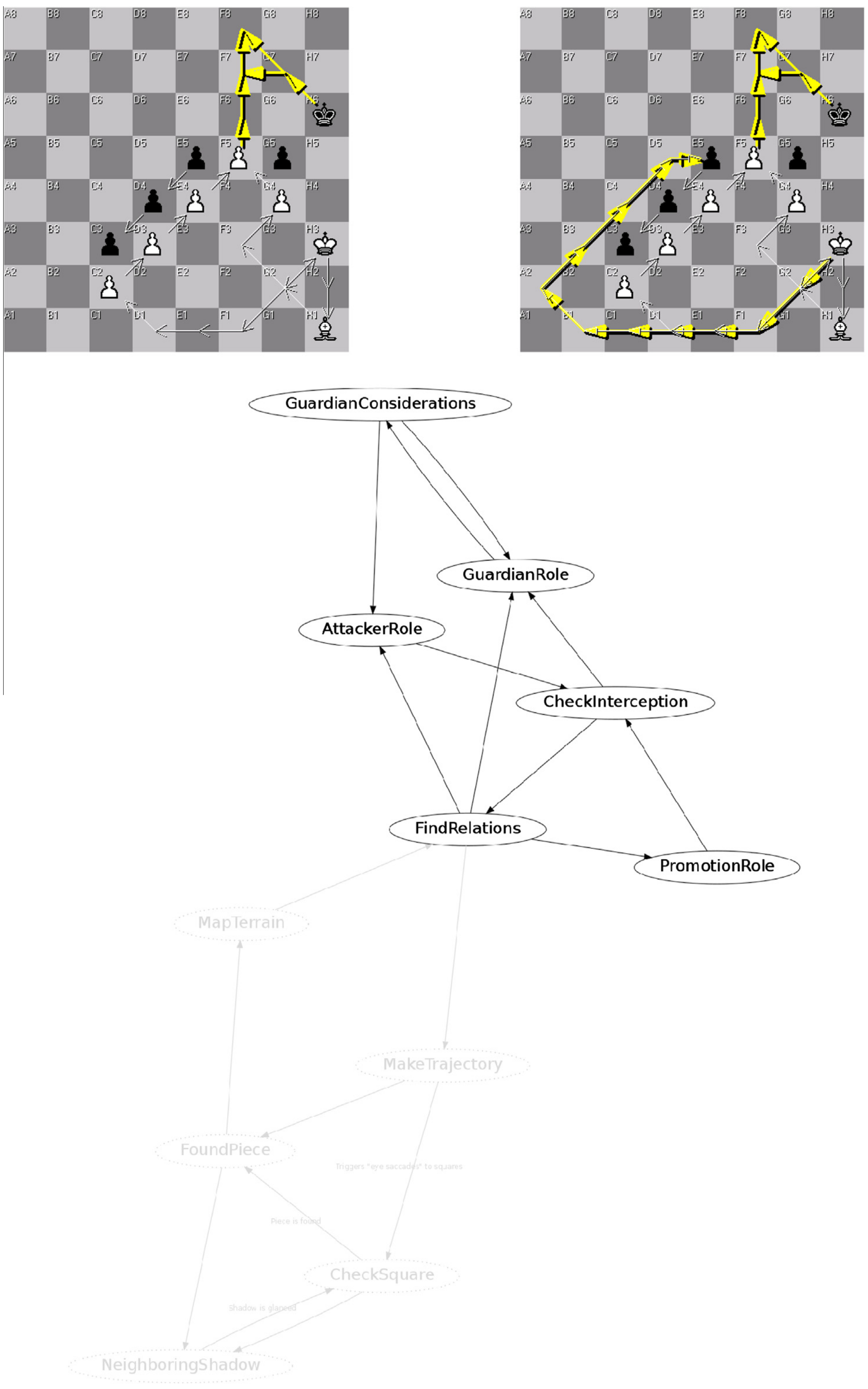


Fig. 11. States 396 and 405.

at these later stages has been triggered mostly by 'expectation-driven,' top-down, codelets: to find new roles for the white side's pieces, for instance, codelets at the trajectory level and piece mobility level are triggered.

After codelet 396 (Fig. 11), all white pieces' 'guardian' roles have very low intensity. For example, intensity of white bishop's defense of the g4 pawn is less than 1/10,000 of the intensity of the potential pawn promotion—the reader should note that an arrow's thickness display is affected by orders of magnitude. Hence, the bishop is seen at this point as a piece of low consequence, with its constrained mobility and nothing to attack in its 'spheres of influence'.

Nine codelets afterwards, at state 405, the white king—finally free from having to guard from any impending attacker—rediscovers the potential attack towards the D5 pawn. This time, however, there is knowledge of the black king's attachment to its role of guardian from the promotion, and the piece cannot assume a second simultaneous role of preventing the white king's attack. It is at this point that Capyblanca registers the fact that the joint attack by the white king and the passed pawn cannot be defended single-handedly by the black king. Intensity of the white king's attack soars to a previously unforeseen level. Hence white's advantage becomes clear, and a strategy to secure that advantage has emerged. The system has achieved, in a mere 405 codelets, an understanding of the global scenario. (The execution continues for 300 codelets with this configuration firmly stable—the system stops after some time with no improvement.)

Some observations may be drawn here:

(i) The architecture is rapidly drawn into the 'relevant' parts of the situation, and attention to irrelevant parts is scarce;
(ii) The emergent process negotiates between top-down (expectation-driven) and bottom-up (data-driven) processes, and does not fall prey to a combinatorial explosion;
(iii) The system is able to gradually move from low-level perception tasks (e.g., a particular piece on a particular square), to higher-level, abstract roles and high-level heuristics that may be applied to those.
(iv) Strategically similar chess positions can be explained *from the point of view of sets of abstract roles*. Their shared essence lies on a high overlap of abstract roles.
(v) For deep understanding of a scenario, the abstraction is *the most efficient level of description*.

The conclusion elaborates with some additional observations.

5. Conclusion

This paper has presented the Capyblanca project, a cognitive model of choice based on the idea that analogy is central to cognition [20,21]. Capyblanca is a multiple-leveled, (simulated) parallel computational architecture, argued to be psychologically plausible (to some extent, obviously). Starting from isolated pieces found in random squares, it gradually converges towards a context-sensitive interpretation of the global scenario. The architecture displays (i) a high degree of 'entropy', continually constructing and destroying structures; (ii) a high degree of parallelism, with both bottom-up and top-down processes running concurrently; (iii) concurrent processing distributed over multiple levels (pieces perceived, distance relations, chess relations, mobility, abstract roles, etc.); and (iv) "vagueness" brought by continuous degrees of intensity of abstract roles. It gradually builds an abstract representation, based on the roles perceived, which gather expectations about what may happen in the position.

How do chess players concentrate attention to relevant aspects of a situation? We propose that players are primed to find relevant information very rapidly, as they perceive the board. Capyblanca models this by placing higher urgencies to (i) processes triggered from new bottom-up information, and (ii) top-down processes with high computed intensity. The architecture is rapidly drawn into the 'relevant' parts of the situation, and attention to irrelevant parts is minimum. There is, however, a constant exploration of alternative "counterfactual" pathways. We believe this wasted effort is psychologically plausible, part of a process of imagining new possibilities, or, in what has been termed in complex systems, exploring "the adjacent possible".

Capyblanca does not compute the chess tree. Instead, the system gradually builds an interpretation of a position, based on various types of information (pieces, trajectories, mobilities, roles, considerations about multiple roles, etc.). It attempts to synthesize information into abstract form: "The black king is not able to defend simultaneously from the passed pawn and the upcoming white king threat". This type of information encoded as "roles" is generalizable (by changing the pieces involved), neither does it concern the accidental trajectories involved. The system, therefore, attempts to find *the essence of the strategic situation*—by manipulating the roles which have been deemed most 'intense'. Instead of a tree-search process, in Capyblanca, elicited cues lead to proper responses, by activating concepts and top-down expectations likely to be relevant to the situation at hand.

Strategically similar chess positions may be explained by looking at the configurations *from the point of view of sets of abstract roles*. For deep understanding, that abstraction is *the most efficient level of description*. While a skeptical reader might consider that the processing of both *trajectories* and of *roles* may be redundantly inefficient, that is really not the case. It is crucial to process *both* trajectories and roles. The distinction is an important one. Because a particular trajectory is bound to a set of squares, it is not an adequate structure to either perceive the situation in abstract terms or to generalize to other superficially different but strategically similar scenarios. Trajectories lead to chess relations, but are different from them. Chess relations are independent of locations and of particular squares of a trajectory. And chess relations are subtly different from abstract roles: In Capyblanca's evolving interpretation of a position, roles emerge gradually from the set of chess relations, by the process of continuous re-evaluation of the intensity of each relation.

This is a distinction, not an argument. By structuring information in terms of abstract roles it is possible to obtain *abstract representational invariance*. Consider, for example, the notion that, "under a double check, the only response is to escape with the king". This notion is generalizable to an immense set of situations. It would be unthinkable—or at least computationally intractable—to try to describe it in terms of particular pieces, particular squares, and particular trajectories. The most efficient level of description is the abstraction "piece A has a role of immediate attacker towards piece X", "piece B also has a role of immediate attacker towards piece X", and "X is a king". It is irrelevant for understanding, and inefficient for processing, what type of piece A and B are, what their mobilities are like, or which squares they occupy. Therefore, it is imperative, for an architecture with the ambition to model abstract vision of a chess position, to use different representational structures to account for the trajectories involved in chess relations, and separate structures for the abstract roles that are found to be relevant in a particular position. Deep understanding arises by recognizing a familiar, previously seen, configuration of abstract roles. For understanding a scenario, the most efficient level of description is given by a combination of the most important roles objects play. By reasoning at this level, a system may be able to perceive analogies between dramatically different scenarios. This enables the eventual emergence of a global, meaningful, understanding of the constraints and key issues underlying the strategic situation one faces.

If Capyblanca were an *ad hoc* system tied exclusively to the chess domain, it would be hardly interesting as a cognitive model. However, it should be emphasized that Capyblanca is one project out of a family of cognitive architectures [5,10–13,20,32,33,35,36,40,41]. It does have severe limitations—which is the key reason why the full source-code is being open-sourced to the community.

As of this writing, the system is, like the original Copycat project, unable to learn [35,36]. Its objective is solely to model human perception and intuitive understanding of a chess scenario "in an individual agent at a particular time" [5]. If one were to ask how the system models chess expertise, the answer is: it does not. Currently, top-down expectations need to be explicitly programmed. The system is unable to acquire new strategies; therefore it cannot play a full game of chess. A full rewrite of the system is underway, which should in theory enable Capyblanca to acquire, during a game, new types of roles, new sets of roles in combinations, and use them in upcoming games. However, the mechanisms for such an architecture have numerous design decisions, and are daunting research problems in their own right. The code for the current version of Capyblanca is open-sourced under the MIT license (a permissive license in which one can fork the code, change the licensing, sell the code or binary executables, but never sue its humble original author). It is hosted at capyblanca.googlecode.com. Capyblanca has been developed using object-oriented pascal and compiles with Turbo Delphi 2005. With this initiative, we hope to contribute to the scientific community by letting those interested not only replicate the results in detail, but also improve the architecture, or fork the project and explore different design decisions that we have not been able to. (Historians of computer science might also be interested, as those thousands of lines of source code do, in all probability, redefine the meaning of spaghetti code).

Yet, despite its current limitations, it may be premature to dismiss Capyblanca as not worthy of scientific interest. While we do not intend to have a top-grade chess engine, *we want to better understand the emergence of choice and the essence of the strategic situation. Chess is merely a domain to explore those ideas more deeply*. Capyblanca may provide a glimpse of the "*devil [...] in the details for how to model a flexible control structure necessary to mediate between potentially competing candidate-move generators driven by top-down analogy processes and bottom-up chunks and templates during a search process.*" [7, p. 323; see also the debate in 15,23,29,31,34,38]. Four years in development, Capyblanca may be, perhaps, a starting point in fully addressing this goal. The general theoretical principles given here should be able to account for expertise; as the higher the skill, the more relevant abstract roles encoded, the greater the system's understanding. Capyblanca may account for more aspects of human thought in chess than alternative models, if indeed it is true that, after the opening of a game, analogy lies at the core of abstract vision.

Acknowledgements

This work was designed and partly conducted while the author was visiting scientist at the Center for Research on Concepts and Cognition of Indiana University, Bloomington; a visit for which the author thanks the memorable discussions with Douglas Hofstadter, Harry Foundalis, Eric Nichols, the late Helga Keller, and the whole FARG team.

Appendix A. Commented code of `GuardianConsiderations` codelet

The interested reader might want to consider how a codelet can deal with abstract reasoning, detached from the low-level perception processes. For this purpose we provide the source code of the `GuardianConsiderations` codelet (note that this project is completely open-source under the liberal MIT license, available at https://code.google.com/p/capyblanca/, and this Appendix intends to only provide a glimpse of its reasoning on a high-level of abstraction).

The crucial logic involved in this codelet refers to pieces assigned roles as "guardians" (of other pieces, or of strategic empty squares in the board). Note that the registration of a guardian is independent of piece type (bishop, pawn, king, etc.), of piece color, of "whatever it happens to be guarding", etc. There are basic two basic "considerations" after the registration that a piece is a "guardian" of "something":

- *Do I really need to be guarding "this"?* If there is another piece, preferably lower ranked in the dynamics of a position, perhaps the role of guardian can be lifted and the piece may be free to assume other roles. In normal circumstances, a pawn is preferable as a guardian than a queen, for example.
- *Is "anyone" else guarding "this"?* If there is no other piece guarding "this", the guardian may not "want" to assume other roles. This is achieved by restraining the guardian's 'sphere of influence' to the squares in which it safely "guards this", wherever these squares may be, and whatever "this" may refer to—see the calls to Guardian.forbid_square().

```
Constructor TGuardianConsiderations.create (P:TParamGuardianConsiderations; U:TUrgency);
begin
     Parameters:=P;
     Urgency:= U;
     Feel_Urging_Pressure (Parameters.impulsepointer);
end;

Procedure TGuardianConsiderations.fire; {Considerations are bound to something}
begin
     With PARAMETERS DO
     BEGIN
     if Node.activation>10 {threshold} then
     begin
          {PART I: Decay}
          Node.Decay;

          Do_I_Really_Need_To_Guard_This;

          {TRIGGER ITSELF ONCE AGAIN}
          Node.Considerations;
     end
     else begin  {CONCEPT HAS DECAYED}
               node.activation:=0;
          end;
     END;
end;

Procedure TGuardianConsiderations.Anyone_Else_Guarding_this;
var x, index:integer; originsquare1, destsquare1, aux_sq: ^tsquare;
    Guardian1, OtherGuy, ImaginedPiece: Tpiece;  Interceptors, Forbidden: Tlist;

begin
     {Step1: someone else lower ranked already guarding it?}
     {1a. scans pieces with same destination square}
     destsquare1:=parameters.trajectory.items[parameters.trajectory.count-1];
     originsquare1:= parameters.trajectory.items[0];
     GUardian1:= workingmemory.GetPieceAt(originsquare1^);

     index:=-1;
     for x:=0 to workingmemory.pieces.Count-1 do
     begin
          OtherGuy:=workingmemory.pieces.items[x];
          if ((OtherGuy<>Guardian1) and (otherGuy.White=Guardian1.White) and (OtherGuy.isDefending(destsquare1^)) and (guardian1.isDefending(destsquare1^)) ) then
               index:=x;
     end;

     if index=-1 then  {nobody is helping to defend destsquare, so...}
     begin
          Interceptors:= GetInterceptors (parameters.trajectory, guardian1);
          if (Interceptors.count>0) then
          for index:=0 to interceptors.Count-1 do
          begin
```

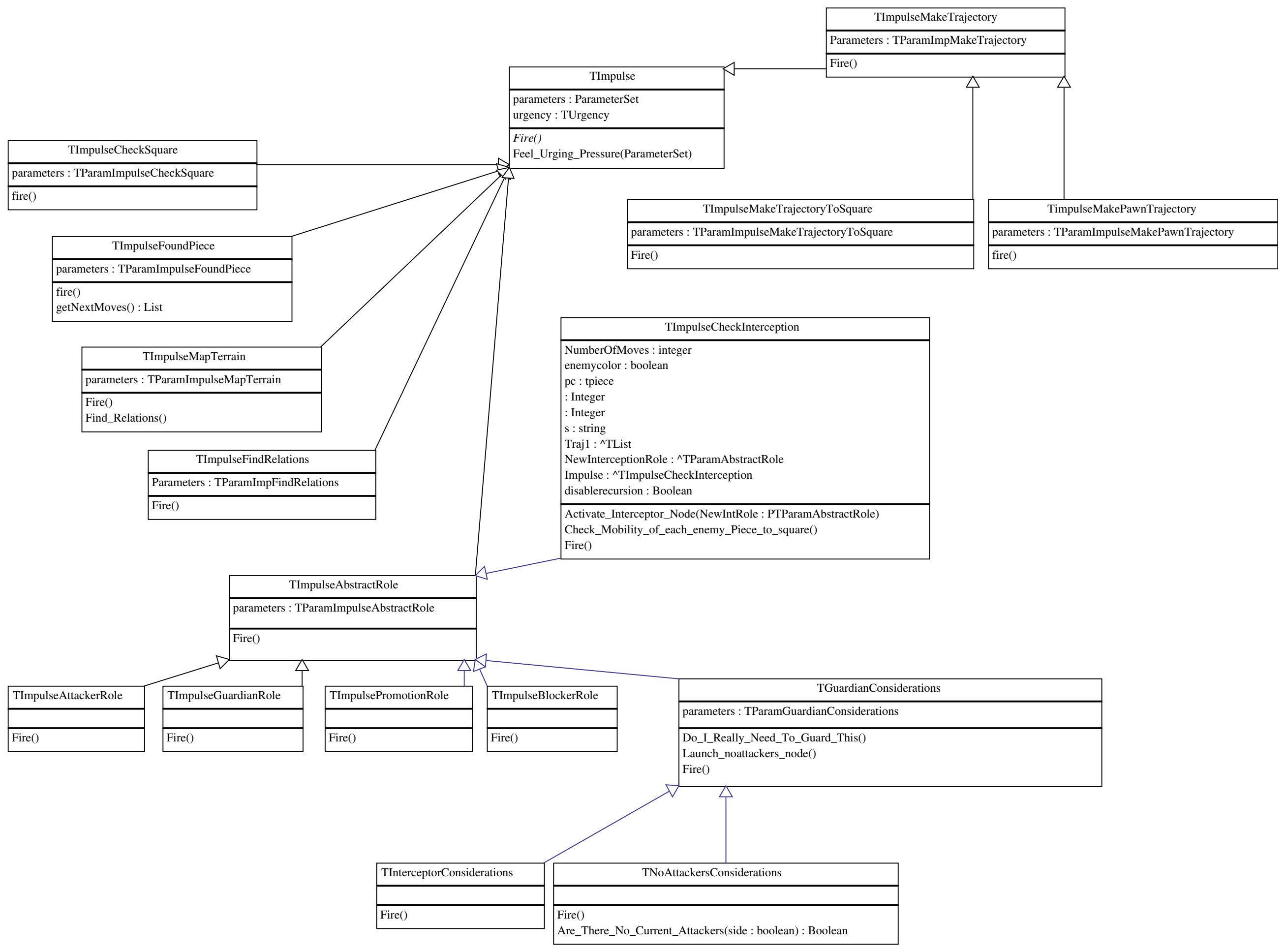
(*continued on next page*)

```
               GUardian1:=Interceptors.items[index];

               Parameters.pmemo.lines.add(Guardian1.fullid+' is totally alone at protecting '+SquareName(destsquare1^));

               Launch_noattackers_node;

               ImaginedPiece:=TPieceFactoryMethod(destsquare1^, Guardian1.White, Guardian1.PieceType);

               Forbidden:= ImaginedPiece.Forbidden_Squares(parameters.Trajectory.Count);
               for x:= 0 to forbidden.Count-1 do
               begin
                    aux_sq:=forbidden.items[x];
                    Guardian1.Forbid_square(aux_sq^);
               end;
          end;
     end;
end;


Procedure TGuardianConsiderations.Do_I_Really_Need_To_Guard_This;
var x:integer; originsquare1, destsquare1: ^tsquare;
    Guardian1, OtherGuy: Tpiece;

begin
     if (parameters.Trajectory.Count>0) then
     begin
     {Step1: someone else lower ranked already guarding it?}
     {1a. scans pieces with same destination square}
     destsquare1:=parameters.trajectory.items[parameters.trajectory.count-1];
     originsquare1:= parameters.trajectory.items[0];
     GUardian1:= workingmemory.GetPieceAt(originsquare1^);
     for x:=0 to workingmemory.pieces.Count-1 do
     begin
          {if found, checks whether piece is lower ranked}
          OtherGuy:=workingmemory.pieces.items[x];
          if ((OtherGuy<>Guardian1) and (otherGuy.White=Guardian1.White) and (OtherGuy.isDefending(destsquare1^)) and (guardian1.isDefending(destsquare1^)) ) then
          begin
               parameters.pmemo.lines.add('    -->'+Guardian1.fullid+ ' is defending something defended by '+OtherGuy.FullId );
               if (otherguy.DynamicValue<Guardian1.DynamicValue) then
               begin
                    {Lower Ranked? Great, exchange intensity with my minimum one / OR / lower intensity to minimum possible
                    / AND / SAY I CAN GO SOMEWHERE ELSE=ROLE=FREE_RIDER}
                    Guardian1.liberate_from_role(destsquare1, parameters.pmemo);
                end else
                begin
                     parameters.pmemo.lines.add('    -->'+OtherGuy.fullid+ ' is already defending something defended by '+Guardian1.FullId );
                     if (otherguy.DynamicValue>Guardian1.DynamicValue) then
                     begin
                          {Higher Ranked? Great, liberate OtherGuy!  CAN GO SOMEWHERE ELSE;ROLE=FREE_RIDER}
                          OtherGuy.liberate_from_role(destsquare1, parameters.pmemo);
                     end

                end;
          end;
     end;
     end;
end;
```

Appendix B. Classes and class hierarchy involved in Capyblanca

This appendix presents to the interested reader the detailed class hierarchy involved in the codelets of this evolving version of Capyblanca (see capyblanca.googlecode.com). Object-oriented Pascal traditionally names classes with the 'T' prefix (for *type*), and the current code also includes the prefix 'Impulse', to distinguish codelet classes from other program classes.

Some of the classes have their own parameter setting structures, and some inherit from parent classes. As noted in Section 3.1, as soon as one of these codelets are launched, they remain in a wait state, modeling potential urges that the system holds, and only after selection, they are launched, through the `Fire()` method. Once again, the reader is referred to the website where the computer code is open-sourced.


TImpulseMakeTrajectory
Parameters : TParamImpMakeTrajectory
Fire()
TImpulse
parameters : ParameterSet
urgency : TUrgency
Fire()
Feel_Urging_Pressure(ParameterSet)
TImpulseCheckSquare
parameters : TParamImpulseCheckSquare
fire()
TImpulseMakeTrajectoryToSquare
parameters : TParamImpulseMakeTrajectoryToSquare
Fire()
TimpulseMakePawnTrajectory
parameters : TParamImpulseMakePawnTrajectory
fire()
TImpulseFoundPiece
parameters : TParamImpulseFoundPiece
fire()
getNextMoves() : List
TImpulseCheckInterception
NumberOfMoves : integer
enemycolor : boolean
pc : tpiece
: Integer
: Integer
s : string
Traj1 : ^TList
NewInterceptionRole : ^TParamAbstractRole
Impulse : ^TImpulseCheckInterception
disablerecursion : Boolean
Activate_Interceptor_Node(NewIntRole : PTParamAbstractRole)
Check_Mobility_of_each_enemy_Piece_to_square()
Fire()
TImpulseMapTerrain
parameters : TParamImpulseMapTerrain
Fire()
Find_Relations()
TImpulseFindRelations
Parameters : TParamImpFindRelations
Fire()
TImpulseAbstractRole
parameters : TParamImpulseAbstractRole
Fire()
TImpulseAttackerRole
Fire()
TImpulseGuardianRole
Fire()
TImpulsePromotionRole
Fire()
TImpulseBlockerRole
Fire()
TGuardianConsiderations
parameters : TParamGuardianConsiderations
Do_I_Really_Need_To_Guard_This()
Launch_noattackers_node()
Fire()
TInterceptorConsiderations
Fire()
TNoAttackersConsiderations
Fire()
Are_There_No_Current_Attackers(side : boolean) : Boolean

References

[1] G. Atkinson, Chess and Machine Intuition, Ablex Publishing, Norwood, NJ, 1993.
[3] Y. Björnsson, T.A. Marsland, Learning extension parameters in game-tree search, Information Sciences 154 (2003) 95–118.
[4] C. Camerer, Behavioral Game Theory: Experiments on Strategic Interaction, Princeton University Press, Princeton, 2003.
[5] D.J. Chalmers, R.M. French, D.R. Hofstadter, High-level perception, representation, and analogy: a critique of artificial intelligence methodology, Journal of Experimental and Theoretical Artificial Intelligence 4 (3) (1992) 185–211.
[6] W.G. Chase, H.A. Simon, Perception in chess, Cognitive Psychology 4 (1973) 55–81.
[7] N. Charness, Patterns of theorizing about chess skill – commentary on Linhares and Freitas (2010) and Lane and Gobet (2011), New Ideas in Psychology 30 (2012) 322–324.
[8] N. Charness, E.M. Reingold, M. Pomplun, D.M. Stampe, The perceptual aspect of skilled performance in chess: evidence from eye movements', Memory & Cognition 29 (2001) 1146–1152.
[9] H.R. Ekbia, Artificial Dreams: The Quest for Non-Biological Intelligence, Cambridge University Press, 2008.
[10] H. Foundalis, PHAEACO: A Cognitive Architecture Inspired by Bongard's Problems. Ph.D. thesis, Indiana University, Bloomington, 2006.
[11] R.M. French, The Subtlety of Sameness, MIT Press, 1995.
[12] R.M. French, When coffee cups are like old elephants or Why representation modules don't make sense, in: A. Riegler, M. Peschl (Eds.), Proceedings of the International Conference New Trends in Cognitive Science, Austrian Society for Cognitive Science, 1997, pp. 158–163.
[13] R.M. French, P. Anselme, Interactively converging on context-sensitive representations: a solution to the frame problem, Revue Internationale de Philosophie 3 (1999) 365–385.
[14] F. Gobet, Expert memory: a comparison of four theories, Cognition 66 (1998) 115–152.
[15] F. Gobet, P.C.R. Lane, S. Croker, P.C.-H. Cheng, G. Jones, I. Oliver, J.M. Pine, Chunking mechanisms in human learning, Trends in Cognitive Sciences 5 (2001) 236–243.
[16] A.D. de Groot, Thought and Choice in Chess, Mouton, New York, 1965.
[18] D.R. Hofstadter, Gödel, Escher, Bach: an Eternal Golden Braid, Basic Books, New York, 1979.
[19] D.R. Hofstadter, Metamagical Themas, Basic Books, New York, 1985.
[20] D.R. Hofstadter, FARG, Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought, Basic Books, New York, 1995.
[21] D.R. Hofstadter, Epilogue: analogy as the core of cognition, in: D. Gentner, K.J. Holyoak, B.N. Kokinov (Eds.), The Analogical Mind: Perspectives from Cognitive Science, MIT Press, 2001.
[22] G. Klein, Sources of Power: How People Make Decisions, MIT Press, Cambridge, MA, 1999.
[23] P.C.R. Lane, F. Gobet, Perception in chess and beyond: commentary on Linhares and Freitas (2010), New Ideas in Psychology 29 (2011) 156–161.
[24] A. Linhares, A glimpse at the metaphysics of Bongard problems, Artificial Intelligence 121 (2000) 251–270.

[25] A. Linhares, An active symbols theory of chess intuition, Minds and Machines 15 (2005) 131–181.
[26] A. Linhares, Dynamic sets of potentially interchangeable connotations: a theory of mental objects, Behavioral and Brain Sciences 31 (2008) 389–390.
[27] A. Linhares, P. Brum, Understanding our understanding of strategic scenarios: what role do chunks play?, Cognitive Science 31 (6) (2007) 989–1007
[28] A. Linhares, P. Brum, How can experts see the invisible?, Reply to Bilalic and Gobet, Cognitive Science 33 (2009) 748–751
[29] A. Linhares, D. Chada, What is the nature of the mind's pattern-recognition process?, New Ideas in Psychology 31 (2013) 108–121
[30] A. Linhares, A.E.T.A. Freitas, Questioning Chase and Simon's (1973) "perception in chess": the "experience recognition" hypothesis, New Ideas in Psychology 28 (2010) 64–78.
[31] A. Linhares, A.E.T.A. Freitas, A. Mendes, J.S. Silva, Entanglement of perception and reasoning in the combinatorial game of chess: differential errors of strategic reconstruction, Cognitive Systems Research 13 (2012) 72–86.
[32] J. Marshall, Metacat: A Self-Watching Cognitive Architecture for Analogy-Making and High Level Perception. PhD thesis, Indiana University, Bloomington, 1999.
[33] G. McGraw, Letter Spirit (part one): Emergent High-Level Perception of Letters Using Fluid Concepts. PhD Thesis, Indiana University, Bloomington, 1995.
[34] S.J. McGregor, A. Howes, The Role of attack and defense semantics in skilled player's memory for chess positions, Memory and Cognition 30 (2002) 707–717.
[35] M. Mitchell, Analogy-Making as Perception, MIT Press, Cambridge, 1993.
[36] M. Mitchell, D.R. Hofstadter, The emergence of understanding in a computer model of concepts and analogy-making, Physica D 42 (1990) 322–334.
[38] D.G. Myers, Intuition: Its Powers and Perils, Yale University Press, New Haven, 2002.
[39] Y. Neuman, O. Nave, Metaphor-based meaning excavation, Information Sciences 179 (2009) 2719–2728.
[40] J.A. Rehling, Letter Spirit (Part Two): Modeling Creativity in a Visual Domain. PhD Thesis, Indiana University, Bloomington, 2001.
[41] J. Silva, A. Linhares, Cognitive Reflection: The 'Premature Temperature Convergence' Hypothesis, in Proceedings of the Twenty-Ninth Conference of the Cognitive Science Society, Nashville, USA, 2007.
[42] D. Wilkins, Using patterns and plans in chess, Artificial Intelligence 14 (1980) 165–203.

Further reading

[2] J. Ban, The Tactics of Endgames, Dover, 1997.
[17] J. Hawkins, On Intelligence, Holt Paperbacks, 2005.
[37] G. Moore, Cramming more components onto integrated circuits, Electronics 38 (8) (1965). <ftp://download.intel.com/museum/Moores_Law/Articles-Press_Releases/Gordon_Moore_1965_Article.pdf> (April 19).