Countdown Numbers Game: Solved, Analysed, Extended

Simon Colton[1]

Abstract. The Countdown Numbers Game is a popular arithmetical puzzle which has been played as a two-player game on French and British television weekly for decades. We have solved this game in the sense that the optimal solution for the nearly 12 million puzzle instances has been generated and recorded. We describe here how we have achieved this using the HR3 Automated Theory Formation system. This has allowed us to analyse the space of puzzles; suggest gamesmanship tactics and game design improvements to the online/handheld versions of the game; and begin to investigate the potential for automatic invention of such games.

1 Introduction

The French television show *Des Chiffres et des Lettres* is one of the longest running quiz shows worldwide, having been on air for 48 years. The British counterpart is called *Countdown*, and is also long running: there have been more than 5000 episodes since its debut in November 1982. Both shows have a section which involves an arithmetical puzzle to be solved by both contestants. In the British version, this is called the *Numbers Game* while in the French version it is called *Le Compte est Bon* ("the total is right"). Each puzzle instance involves an **input list** which is a randomly ordered sublist of 6 elements from this integer list: $\{1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 10, 10, 25, 50, 75, 100\}$ The numbers 1 to 10 are called the **small numbers**, with the numbers 25, 50, 75 and 100 being called the **large numbers**. Contestants apply only the basic arithmetical operators (addition, subtraction, multiplication and division) to the inputs to arrive at a randomly chosen **target number**. Each input number may be employed only once in the **solution**, and no fractional or negative numbers can be employed. There is no requirement to use all the input numbers.

The British and French versions differ a little. In the British version, the target number is between 100 and 999 and contestants are given 30 seconds to solve the puzzle, while in the French version, the target is between 101 and 999, and the time limit is 45 seconds. In both cases, the target number is calculated using a random number generator which is not linked to the input list. In the French version, the input numbers are chosen randomly by computer, whereas in the British version, a contestant chooses from the shuffled integer list. The choice is blind, but the contestant has the option to include 0, 1, 2, 3 or 4 of the large numbers. There are two Numbers Games in each show, hence both contestants get to choose the numbers for an instance. Scoring of the game also differs slightly between the two versions. In Countdown, contestants score 10 points if they achieve a perfect solution, but if neither achieves this, then the contestant or contestants with the closest **answer** scores 7 points if they are within 5 of the target, and 5 points if they are within 10. In Des Chiffres et des Lettres, contestants score 10 if they get a perfect solution, but if neither achieves this, the contestant or contestants achieving the closest answer scores 7 points.

Figure 1. An example of the Countdown Numbers Game involving all four large numbers and two small numbers as input, with target integer 952.

One solution to this puzzle is:

$((((75*6)/50)*(100+3))+25)$

An example puzzle from Countdown, to which we refer throughout, is given in figure 1. While fairly difficult for most people, solving instances of the puzzle is relatively easy for software, and there is an abundance of online solvers available. Many of these solvers claim to be perfect in the sense that they will always give an optimal solution (with the notion of optimal changing) for any problem instance. For instance, the solver available at: `www.crosswordtools.com/numbers-game` is designed to give the most *intuitive* solution based on the difficulty of applying the different arithmetical operators (e.g., with addition being easier to apply than division). Other aspects of how difficult a puzzle instance might be for a person include the number of inputs required for a solution and the size of the largest number used in the calculation. For instance, when solving the puzzle in figure 1, contestant James Martin calculated $318*75 = 23850$ and $23800/25 = 952$ to find a solution. The simpler solution in figure 1 requires lesser mental feats. The Compte est Bon variant of the puzzle was employed by Defays in [8] and chapter 3 of [9] to study relations between perception and cognition as part of Hofstadter et. al's fluid analogies programme.

Concentrating on the Countdown variant, to solve this in the sense that the 15-puzzle and Rubik's Cube have been solved, means calculating and storing the optimal solution to each puzzle instance. For the Numbers Game, such a total solution can be achieved through generating each problem instance and solving it using a trusted solver. This has been achieved by Alliot in unpublished (in the peer-reviewed sense) work, via a detailed and interesting investigation [1] of the puzzle space, with an emphasis on complexity analysis. Alliot uses a highly optimised solver which uses a breadth first search and is able to solve single instances in mere milliseconds. He reports that it solves the entire puzzle space in 53 seconds. It is fair to say that this approach frames the task of solving the Countdown Numbers game in the *problem solving paradigm of AI*, as discussed in [6], whereby an intelligent task to perform is interpreted as a series of problems to be solved. Our approach is different. As described below, we have framed the task within the *artefact generation paradigm of AI*, whereby intelligent tasks are interpreted as a series of valuable objects to be generated. Our approach is slower, as it exhausts the space for solutions for every puzzle instance, but there are benefits to having all solutions, as discussed later.

[1] Computational Creativity Group, Department of Computing, Goldsmiths, University of London `ccg.doc.gold.ac.uk`

The advantage to solving games is summarised neatly in [10]: "Solving a game takes this to the next level by replacing the heuristics with perfection". In addition to always providing perfect answers to given puzzles, puzzles more appropriate to ability can be selected when a game has been solved, as the puzzle space can be analysed and aspects of its nature determined and utilised, as for the Numbers Games in section 4. In addition, we can use such projects to investigate and validate the abilities of a generic AI approach in a novel situation. To this end, we have solved the Numbers Game using the HR3 Automated Theory Formation system, which is described in section 2, with details of its application given in section 3. We conclude by discussing the potential of automatically inventing new games, by briefly describing a novel variant of Countdown that we have solved.

2 The HR3 System

Automated Theory Formation (ATF) is a hybrid AI approach which starts with minimal background knowledge about a domain and produces a *theory*. Such a theory consists of examples, concepts which categorise the examples, conjectures which relate the concepts, and proofs which demonstrate the truth of certain conjectures, which become known as theorems. The first implementation of an ATF system was HR1 [4], written in Prolog, and the second, Java, implementation was HR2 [5]. Both systems have been used for mathematical discovery tasks, some of which are summarised in [5], in addition to artefact generation projects in non-mathematical domains.

The latest implementation, the HR3 system described in [7], has been engineered with both speed and memory efficiency in mind: it can run up to 15000 times faster than HR2 and can store millions of concepts within a modest memory capacity. Providing full details of how HR3 works is beyond the scope of this paper. Of importance here is the fact that it uses *production rules* to take one (or two) old concepts and generate the examples of a new concept from them. There are currently 13 production rules, but more will be added (HR2 has 22). These are split into: 9 logical rules which, for instance, manipulate concepts by introducing universal or existential quantification, and use composition, negation, etc., and 4 numerical rules which manipulate the numbers in concepts, e.g., with numerical relations and by counting subobjects. In particular, HR3 has an *Arithmetic* production rule which is able to take two concepts which contain numerical information and apply addition, subtraction, multiplication and division to the numbers. This is the only production rule we required for the Countdown application.

For improved efficiency, unlike HR2, much of HR3's functioning is *on-demand*. For instance, it does not generate a definition for a newly generated concept until the user asks for one. HR3 does record the *construction history* of each concept, which enables the data for it to be constructed from scratch from the background knowledge, and definitions can be similarly generated in this fashion. Moreover, for efficiency, there are a number of ways in which HR3's search for concepts can be curtailed. Firstly, each production rule has the ability to refuse to apply itself to certain input concepts, and we describe how we set up the Arithmetic production rule in this respect in the next section. Secondly, general or bespoke analysis modules can be used at runtime to (i) stop a production rule step (or whole sets of steps) being carried out in advance and (ii) refuse to allow a newly invented concept into the theory, hence reducing the search, as no concepts will be built from it later. Again, in the next section, we describe how we introduced two modules in order to break symmetries in the theory formation sessions for the Countdown application, drastically improving efficiency.

3 Solving the Countdown Numbers Game

We define a puzzle *instance* as a quadruple $P = \langle T, I, C, A\rangle$, where T is the target number, I is the list of six input numbers ordered from smallest to largest, C is the calculation performed over I, and A is the answer, or result of the calculation (which may or may not be the same as T). Note that the calculation C must follow the rules, i.e., involving each $i \in I$ only once and requiring the calculation of no negative or fractional numbers at any stage. For a given instance P, we denote P_T to be the target of P, with P_I, P_C and P_A defined similarly. It is clear that this covers all the possible puzzles which could be shown on the television show up to permutation of the input numbers, and there is no need to consider puzzles which do not have numerical ordering on the inputs, as these are trivially isomorphic to an instance as defined above.

An instance P is said to be *correct* if $T = A$ or $\nexists\, P'$ s.t. $P' = \langle T, I, C', A', \rangle$ is an instance, and $|T - A'| < |T - A|$. For an instance P, we define $U(P)$ to be the number of input numbers used in P_C, $L(P)$ to be the largest number calculated at any intermediary stage of P_C, not including A, and $N(P)$ to be the number of distinct numerical operators used (counted only once) in the calculation. Given two instances P and Q, we denote $P < Q$ if $P_T = Q_T, P_A = Q_A$ and either (i) $U(P) < U(Q)$ or (ii) $U(P) = U(Q)$ and $L(P) < L(A)$ or (iii) $U(P) = U(Q)$ and $L(P) = L(A)$ and $N(P) < N(Q)$. We say that an instance P is *optimal* if $\nexists\, P'$ s.t. $P' < P$. Note that for any pair $\langle T, I\rangle$, there may be multiple optimal instances, and that this formulation of optimality is justified by the discussion in section 1, but arbitrary, and we plan to investigate other candidates in future work, in particular those more related to how people solve puzzle instances.

As an example, in figure 1, the puzzle is an isomorphism of this correct instance:

$P = \langle 952, \{3, 6, 25, 50, 75, 100\}, ((((75*6)/50)*(100+3))+25), 952\rangle$

Moreover, $U(P) = 6$, as all six inputs are used in P_C, and $L(P) = 927$ because $((75*6)/50)*(100+3) = 927$ is the intermediary calculation resulting in the biggest number. The contestant in the game shown in figure 1 did very well to find the instance:

$Q = \langle 952, \{3, 6, 25, 50, 75, 100\}, (((100 + 6) * 3) * 75) - 50)/25), 952\rangle$

but this is not optimal in our sense, because $P < Q$, given that $U(P) = U(Q)$ and $L(P) = 927 < L(Q) = 23850$. On the YouTube site[2] showing the TV clip of this game, one of the commentators has pointed out this easier solution.

3.1 Setting Up and Running HR3

Given the above definitions, solving the Countdown Game puzzle in its entirety involves finding a correct, optional instance $\langle T, I, C, A\rangle$ for every pair $\langle T, I\rangle$ allowed under the rules. As an exercise in automated theory formation, we first attempted to get HR3 to find every optimal instance in a single run. This nearly worked, but it exceeded a memory limit – we plan to come back to this approach in future work. Instead, we took each of the possible lists of numerically ordered six input numbers I as the background knowledge to a theory formation session and generated all possible calculations (up to isomorphism) involving a subset of I. From these, we determined all optimal instances of Countdown puzzles, as described below.

Given an input list $I = \{x_1, \ldots, x_6\}$, each x_i was used to produce the background concept of 'being the particular instance of number x_i'. Concepts input to HR3 are normally more like prime numbers,

[2] www.youtube.com/watch?v=6mCgiaAFCu8

which have more than one example. However, giving each x_i its own concept (with only one example, naturally) enabled us to break symmetries in the search space and drastically increase efficiency, as described below. Moreover, for input sets where there is a repeated integer, by giving each of the pair its own concept, we were able to tidy up the processing enormously (details omitted). Starting from the background information, the Arithmetic production rule was used iteratively to take pairs of concepts and calculate new concepts like the concept of singletons X such that $X = 10 + 25$, and from this, $X = (10 + 25)/5$, etc. We told HR3 to stop after all possible calculations involving all the x_i had been exhausted, which required five applications of the Arithmetic production rule. Calculations involving all the x_i such as $(a+b)*(c+d)*(e+f)$ were generated after only three applications, because $(a + b)$, $(c + d)$ and $(e + f)$ were all generated with the first application. However, calculations such as $(a * (b/(c + (d - (e + f)))))$ only came out after five applications.

In addition to isomorphism up to permutation of the input numbers, two puzzle instances may be isomorphic in terms of the calculation, C, performed. This is due to the commutativity of the addition and multiplication operators. Hence, we set flags for the Arithmetic production rule which allowed the addition and multiplication of concept C_1 with C_2, but not C_2 with C_1. Given that no negative numbers are allowed at any stage in the calculation, we also ruled out any application of subtraction where the right hand number was greater than the left hand number. Also as the rules don't allow it, we similarly ruled out any calculation resulting in a fractional value. We also ruled out any subtraction which results in zero. This doesn't remove optimal instances from the results, because adding and subtracting zero doesn't change the calculation, division by zero is ruled out by the rules of simple arithmetic, and multiplication by zero leads to a result of zero, and hence there is a simpler solution which involves writing nothing. We ruled out multiplication by 1 and subtraction/addition of 0 for similar reasons. Additionally, before the final application of the Arithmetic production rule, it was told to rule out any calculations resulting in an answer of less than 90 or greater than 1009, as these would not be of use.

Even with these symmetry breaking constraints on the functioning of the Arithmetic production rule, there was still much redundancy in the search space to be removed. Firstly, using an existing module, we told HR3 not to combine any pairs of concepts which contain a shared background concept in their construction history, as this would represent a calculation using an integer from the input set twice, which is not allowed. Also, suppose the concept C_1 performing the calculation $8 = ((3 + 1) + 4)$ had already been generated by HR3, and then later it invented concept C_2 which calculated 8 in a different way using the same numbers, e.g., $8 = (3 - 1) * 4$. Note that the former uses only addition, while the latter uses subtraction and multiplication, and recall that we are interested in generating the *optimal* Countdown instances, as defined above. It is clear that any instance, P containing the sub-calculation $((3 - 1) * 4)$ would not be optimal, because there would be at least one instance Q containing the sub-calculation $((3 + 1) + 4)$ for which $Q < P$, hence any calculations based on subcalculations $((3 - 1) * 4)$ and the like are ruled out, along with calculations such as $18 = (5 * 4) - 2$ in favour of $18 = (5 + 4) * 2$ due to a smaller intermediate calculation, and $((5 * 4) - (3 - 2))$ in favour of $((5 * 3) + 4)$ because of the smaller number of inputs used. To make sure no optimal solutions were lost, if the new concept was better than the existing one (in the optimality sense), then the existing one is substituted with the new one.

Once the theory was produced, we employed a bespoke module to go through it and for each target number, n, between 100 and 999, find the optimal calculation which achieved n and record the number of sub-optimal calculations which also achieve n. In the cases where it was not possible to achieve n exactly, the module found the calculation resulting in $n \pm k$ for k as small as possible, choosing $n-k$ when $n-k$ and $n+k$ were equal in terms of optimality. To increase efficiency, we distributed the theory formation sessions over a multi-threaded machine. We used a Dell server with four processors each able to run 32 parallel threads at 2.9 Ghz. We tried various load balancing setups and found that distributing the sessions as shell processes (calling HR3 with the input numbers given as command line arguments) randomly over the four machines was the most efficient.

The number of different ordered lists of six integers taken from the Countdown possibilities is 13243, as calculated at `www.crosswordtools.com/numbers-game/faq.php`, and this concurred with our generation of all possible background theories for HR3. Hence, given the integers 100 to 999 as targets, there are $13243 * 900 = 11918700$ different puzzles to solve. With the parallel setup above, HR3 took 1771 seconds, or 29.5 minutes to generate optimal solutions to every problem instance. Given that 128 threads were running concurrently, the average duration of a theory formation session (which accounted for 900 instances) was therefore $(1771/13, 243) * 128 = 17.12$ seconds, hence it solved individual puzzle instances in 19ms on average. In preliminary testing, leaving out any of the symmetry breaking techniques, resulted in theory formation sessions for a given input list lasting tens of minutes, which would have ruled out performing all the sessions.

4 An Analysis of the Puzzle Space

Table 1 provides an analysis of the space of puzzles in the Countdown Numbers Game. This was calculated from the solved instances, which are stored on file as tuples of the form:

$\langle T, A, D, I, C, A, U(P), N(P), L(P), S_1(P), \ldots, S_6(P), E\rangle$

where $\langle T, I, C, A\rangle$ defines an instance, P, $D = |T - A|$, $S_i(P)$ is the number of calculations using i inputs from I that achieve A, and E is an explanation of the calculation as list of sub-calculations. For instance, the explanation for the example in figure 1 is: *5*6=450, 450/50=9, 100+3=103, 9*103=927, 927+25=952*. The columns in table 1 break the space down in terms of the number of big numbers in the input list, with the first column of results representing the whole puzzle space. The rows give various raw numbers and percentages pertaining to the instances within the sub-space of puzzles. In particular, the number of instances overall is broken down into those which are solvable scoring max 10, 7 and 5 points, and those which are unsolvable, given along with the expected score.

We define an instance, P, to be *easy* if $U(P) \leq 3$, *medium* if $U(P) = 4$ and *hard* if $U(P) \geq 5$. We further define an instance to be *isolated* if $S_{U(P)}(P) = 1$, and *difficult* if $P_T = P_A$, $U(P) = 6, L(P) \geq 1000, N(P) = 4$ and $S_6(P) = 1$. To achieve the 10-point perfect solution (which is possible), such difficult problems require the usage of all six numbers and all four operators, and the calculation of a intermediate number greater than or equal to 1000. In addition, they are isolated, i.e., there is only a single way (up to arithmetical isomorphism) to solve such difficult problems. Note that the example in figure 1 (which is celebrated on the internet as a particularly thorny example) is not classed as difficult under this scheme for three reasons: $N(P) = 3 < 4$, $L(P) = 927 < 1000$ and $S_6(P) = 2 > 1$. As per table 4, there are 408515 isolated instances (3.43%) and 8614 difficult instances (0.07%), with the one requiring the highest intermediate calculation (of 99300) being:

$\langle 993, \{1, 3, 25, 50, 75, 100\}, (((((50 + 3) * 25) - 1) * 75)/100), 993\rangle$

In one sense, this is the most difficult Countdown puzzle possible.

Big numbers	≥ 0	≥ 1	0	1	2	3	4	Primedown
Instances	11918700	9353700	2565000	5227200	3321000	756000	49500	7266600
Solvable (10 pts)	10871837 (91.22)	8905413 (95.21)	1966424 (76.66)	4971884 (95.12)	3195793 (96.23)	693971 (91.80)	43765 (88.41)	7126391 (98.07)
Solvable (7 pts)	913165 (7.66)	442114 (4.73)	471051 (18.36)	251637 (4.81)	123925 (3.73)	60969 (8.06)	5583 (11.28)	139292 (1.92)
Solvable (5 pts)	28805 (0.24)	3777 (0.04)	25028 (0.98)	2003 (0.04)	856 (0.03)	792 (0.10)	126 (0.25)	367 (0.01)
UnSolvable (0 pts)	104893 (0.88)	2396 (0.03)	102497 (4.00)	1676 (0.03)	426 (0.01)	268 (0.04)	26 (0.05)	550 (0.01)
Exp. Score	9.67	9.85	9.00	9.85	9.89	9.75	9.64	9.94
Easy	772172 (6.48)	740876 (7.92)	31296 (1.22)	352963 (6.75)	311845 (9.39)	72314 (9.57)	3754 (7.58)	642692 (8.84)
Medium	3209093 (26.92)	2875164 (30.74)	333929 (13.02)	1533042 (29.33)	1110115 (33.43)	221935 (29.36)	10072 (20.35)	2554972 (35.16)
Hard	7832542 (65.72)	5735264 (61.32)	2097278 (81.77)	3339519 (63.89)	1898614 (57.17)	461483 (61.04)	35648 (72.02)	4068386 (55.99)
Difficult	7808 (0.07)	7750 (0.08)	58 (0.00)	1471 (0.03)	2829 (0.09)	3013 (0.40)	437 (0.88)	1305 (0.02)
Isolated	408515 (3.43)	240961 (2.58)	167554 (6.53)	139166 (2.66)	69165 (2.08)	29517 (3.90)	3113 (6.29)	84259 (1.16)
Av. Max. Calc.	353.04	389.88	218.72	348.39	409.07	536.86	1238.37	372.56

Table 1. An analysis of the Countdown Numbers Game space of puzzles. Percentages are given in brackets where appropriate.

The analysis in table 1 matches that of [1], hence the two approaches corroborate each other. HR3's search is similar to the breadth first search used in [1], with one major difference: HR3's search is complete, whereas in [1], each problem solving event stops as soon as a solution has been found. It is difficult to imagine how the problem solving approach could determine the isolated or difficult instances without exhausting the space. Such information would be valuable, if we wanted to present, say, a 'champions' version of the Numbers Game with only the difficult instances (perhaps for so-called *brain training* entertainment purposes). Similarly, if variants of the game were to be investigated, for example one where two completely distinct solutions are required, or all the input numbers have to be included in the solution, or there is a bonus for using the number 17, this would probably require a more exhaustive search.

Somewhat ironically, we can use the computational analysis to highlight how good the Countdown Numbers Game is as a pen and paper past-time. Gifted puzzlers should be rewarded with full points, not held back by the design of the game itself, and given an analysis of the entire puzzle space, we can determine the value of the game. From table 4, we see that for 104893 (0.88%) of the puzzles, no score is possible. Hence, roughly one in a hundred games written down would be futile. However, this risk is mitigated by ensuring that at least one of the big numbers is chosen, as the probability of a futile game reduces to 0.03%. In addition, nearly 6.5% of the games are classed as easy, hence there is often much thumb-twiddling[3] on the television show, as the puzzle is often no challenge for either contestant. The best choice of numbers to reduce this is zero big numbers, as only 1.22% of such puzzles are easy, but then the chances of scoring the full 10 points drastically reduces.

We can also use the analysis to make gamesmanship suggestions. Recall that in the UK game show, contestants can choose 0, 1, 2, 3 or 4 big numbers for the inputs. Stronger players may make different choices than weaker players for strategic reasons, especially if they know the rough ability of their opponent (which is sometimes the case on the game show). For instance, a strong player playing against a weak player might choose zero big numbers, as 81.77% of instances are likely to be too hard for their opponent, while the number of difficult puzzles (perhaps too hard even for a strong player) is negligible. However, they should be aware that their expected score will reduce to 9.00 from 9.67, and there is a 4% chance that they will score zero. At the other end of the spectrum, if – like the contestant in figure 1 – they choose four big numbers, their expected score will remain high, but nearly 1 in 100 puzzles will be difficult, and the maximum intermediate calculation will rocket to 1238 on average.

[3] A commentator on a recent newspaper article [11]: *"They should change the random number thingy so it doesn't come up with a really easy target number, meaning the contestants sit there like stiffs for nearly 30 seconds"*

5 Conclusions and Future Work

We described how HR3 found optimal solutions to each of the nearly 12m Countdown Numbers Game puzzles. With the data we have calculated, there is no need for TV viewers or online players to endure futile (i.e., no scoring solution) or thumb-twiddling (i.e., too easy) events. Of the ten online and handheld Countdown puzzle generators and solvers we found, none could tailor the problem to the ability of the player, and our data would enable such enhancements. In addition to providing an analysis of the Numbers Game, and suggesting enhancements, this has been a suitable test for our software, showing that HR3 is able to contribute in game solving and analysis.

We plan to use HR3 to invent new puzzles in a similar way to how HR2 constructed puzzle instances [3], and Browne's Ludi system invented new and interesting board games [2]. We will investigate sampling the puzzle space of new game designs, rather than solving them entirely, to enable the exploration of more designs, and we will look at different rulesets, mathematical operators and scoring mechanisms. To investigate the potential for this, we invented and solved *'Primedown'*, which replaces the numbers available for the input list with two copies of the prime numbers between 2 and 37 inclusive. As we see from the final column in table 1, as a pen-and-paper game, Primedown has a higher expected score, far fewer futile instances, and a more equal spread over easy, medium and hard puzzles than any variant of Countdown, which we find very encouraging.

Acknowledgements

This work has been supported by EPSRC grants EP/J004049 and EP/I001964, and EC FP7 grant 611553 (COINVENT). Many thanks to the anonymous reviewers for their useful comments.

REFERENCES

[1] J-M Alliot, '(The Final) Countdown', *alliot.fr/COMPTE/compte.html, alliot.fr/papers/compte.pdf*, (2013).
[2] C Browne and F Maire, 'Evolutionary game design', *IEEE Transactions on Computational Intelligence and AI in Games*, **2(1)**, (2010).
[3] S Colton, 'Automated puzzle generation', in *Proceedings of the AISB Symposium on AI and Creativity in the Arts and Science*, (2002).
[4] S Colton, *Automated Theory Formation in Pure Maths*, Springer, 2002.
[5] S Colton and S Muggleton, 'Mathematical applications of Inductive Logic Programming', *Machine Learning*, **64**, (2006).
[6] S Colton and G Wiggins, 'Computational Creativity: The final frontier?', in *Proceedings. of the 20th ECAI*, (2012).
[7] S Colton, R Ramezani and T Llano, 'The HR3 Discovery System', in *Proceedings of the AISB Symposium on Scientific Discovery*, (2014).
[8] D Defays, 'Numbo: A study in cognition and recognition', *J. for the Integrated Study of AI, Cog. Sci. and App. Epistemology,* **7(2)**, (1990).
[9] D Hofstadter, *Fluid Concepts & Creative Analogies*,Basic Books,1995.
[10] J Schaeffer, N Burch, Y Björnsson, A Kishimoto, M Müller, R Lake, P Lu, and S Sutphen, 'Checkers is solved', *Science*, **317(5844)**, (2007).
[11] G Virtue, 'Countdown is 70: Three cheers for the nation's favourite comfort blanket', *Guardian*, (7th January 2014).