Mastermind (SPIN 2011)

AVACS S3
Phase 2

July 28, 2011

1 Description of the Benchmark

In the game of Mastermind [1] one player, the *guesser*, tries to find out a *code*, generated by the other player, the *encoder*. The code consists of a number of tokens of fixed positions, where for each token one color (or other labelling) out of a prespecified set is chosen. Colors can appear multiple times. Each round, the guesser guesses a code. This code is compared to the correct one by the encoder who informs the guesser on

1. how many tokens were of the correct color *and* at the correct place and
2. how many tokens were *not* at the correct place, *but* have a corresponding token of the same color in the code.

Notice that the decisions of the encoder during the game are deterministic, while the guesser has the choice between all valid codes. We assume that the encoder chooses the code probabilistically with a uniform distribution over all options. The guesser's goal is to find the code as fast as possible and ours is to compute the maximal probability for this to happen within t rounds.

2 Formalization of the Game

We formalize the game as follows: we let n be the number of tokens of the code and m the number of colors. This means there are m^n possible codes. Let O denote the set of all possible codes. We now informally describe the I/O-IPCs which represent the game. The guesses are described by actions $\{g_0 \mid o \in O\}$, whereas the answers are described by actions $\{a_{(x,y)} \mid x, y \in [0, n]\}$.

The guesser repeats the following steps: From the initial state, s_G, it first takes a probabilistic step to state s'_G and afterwards the guesser returns to the initial state via one of m^n transitions, each labelled with an output action $g_0!$. In both states the guesser receives answers $a_{(x,y)}?$ from the encoder and for all answers the guesser simply remains in the same state, except for the answer $a_{(n,n)}?$ which signals that the guesser has guessed correctly. When the guesser receives this action it moves to the absorbing state s''_G.

Settings			PMC			*PARAM*		NLP		
n	m	t	$\#S$	$\#T$	$\#V$	Time (s)	Mem (MB)	$\#V$	Time (s)	Pr
2	2	2	197	248	36	0.0492	1.43	17	0.0973	0.750
2	2	3	629	788	148	0.1300	2.68	73	0.6530	1.000
3	2	2	1545	2000	248	0.2760	5.29	93	1.5100	0.625
3	2	3	10953	14152	2536	39.8000	235.00	879	1.4330	1.000
2	3	2	2197	2853	279	0.5090	6.14	100	2.1500	0.556

Table 1: Results of Mastermind Case Study

The encoder E is somewhat more complex. It starts by picking a code probabilistically, where each code has the same probability $\frac{1}{m^n}$. Afterwards the encoder repeats the following steps indefinitely. First it receives a guess from the guesser, then it replies with the appropriate answer and then it takes a probabilistic transition. This probabilistic step synchronizes with the probabilistic step of the guesser, which allows us to record the number of rounds the guesser needs to find the code.

The Mastermind game is the composition $C := G \parallel E$ of the two basic I/O-IPCs. We can now reason about the maximal probability $P(F^{\leq t} s''_G)$ to break the code within a prespecified number t of guesses. We consider here the set of all distributed schedulers as we obviously want that the guesser uses only local information to make its guesses. If we were to consider the set of all schedulers, the maximum probability would be 1 for any time-bound as the guesser would immediately choose the correct code with probability 1. If only two components are considered, every distributed scheduler is also strongly distributed. Therefore we omit the analysis under strongly distributed schedulers for this case study.

3 Results

Results are given in Table 1. In addition to the model parameters (n, m), the time bound (t) and the result (Pr) we provide statistics for the various phases of the algorithm. For the unfolded parametric Markov chain (PMC) we give the number of states $(\#S)$, transitions $(\#T)$ and variables $(\#V)$. For *PARAM* [2], we give the time needed to compute the polynomial, the memory required and the number of variables that remain in the resulting polynomial. Finally we give the time needed for *Matlab* to optimize the polynomial we obtained, under the linear constraints that all scheduler decisions lie between 0 and 1. For this case study we generated PMC models and linear constraints semi-automatically, given the parameters n, m and t.

References

[1] Leslie H. Ault. *Das Mastermind-Handbuch*. Ravensburger Buchverlag, 1982.

[2] Georgel Calin, Pepijn Crouzen, Pedro R. D'Argenio, Ernst Moritz Hahn, and Lijun Zhang. Time-Bounded Reachability in Distributed Input/Output Interactive Probabilistic Chains. In *SPIN*, pages 193–211, 2010.