MRMC

Quantitative Logics

David N. Jansen

MRMC

- Markov Reward Model Checker
- model checker for:
  - Markov chains
  - Markov reward models
- research tool:
  thorough mathematics, no fancy user interface
- authors: Joost-Pieter Katoen (initiator),
  Ivan Zapreev, Maneesh Khattri (most code),
  David N. Jansen (bisimulation), ...

input format

- describe DTMC in a file
  - .tra file: transition matrix
  - .lab file: labelling with atomic propositions
- provide file names on command line
- provide formulas while the tool runs

Formal definition: labelled DTMC

- A Markov chain consists of:
  - S — finite set of states (often $S = \{1, 2, \ldots n\}$)
  - $\mathbf{P}$: $S \times S \rightarrow [0,1]$ — transition probability matrix (with row sums = 1)
  - π_0: $S \rightarrow [0,1]$ — initial state distribution (sometimes)
  - L: $S \rightarrow 2^{AP}$ — labelling with atomic propositions

code walkthrough

- next formulas
- bounded until formulas
- unbounded until formulas

next

```
double * dtmc_unbounded_next(const bitset * phi) {
	const sparse *state_space = get_state_space();
	const int size = mtx_rows(state_space);
	int i;
	double * result = (double *) calloc((size_t) size, sizeof(double));
	double * i_phi = (double *) calloc((size_t) size, sizeof(double));

	i = state_index_NONE;
	while ( (i = get_idx_next_non_zero(phi, i)) != state_index_NONE ) {
			i_phi[i] = 1.0;
	}

	multiply_mtx_MV(state_space, i_phi, result);
	free(i_phi);
	return result;
}
```

bounded until

```
double * dtmc_bounded_until(const bitset *phi, const bitset *psi, double supi) {
	const sparse * state_space = get_state_space();
	int i;
	const int size = mtx_rows(state_space);
	double * result_1 = (double *) calloc((size_t) mtx_rows(state_space),
			sizeof (double));
	bitset *good_phi_states = get_good_phi_states(phi, psi, state_space);
	int * pValidStates = count_set(good_phi_states);
	sparse *pP = allocate_sparse_matrix(size, size);
	makeAbsorbingDTMC(state_space, pP, pValidStates);

	i = state_index_NONE;
	while ( (i = get_idx_next_non_zero(psi, i)) != state_index_NONE ) {
			result_1[i] = 1.0;
	}
	dtmc_bounded_until_universal(pP, &result_1, supi);

	free_bitset(good_phi_states); free(pValidStates); free_mtx_wrap(pP);
	return result_1;
}
```

```
static void dtmc_bounded_until_universal(sparse *pP, double **ppInOutData,
      double supi)
{
      int i;
      double *result_2, *pTmp;
      double * result_1 = *ppInOutData;

      result_2 = (double *) calloc((size_t) mtx_rows(pP), sizeof(double));
      for(i = 1; i <= supi ; i++) {
            multiply_mtx_MV(pP, result_1,result_2);
            pTmp = result_1;
            result_1 = result_2;
            result_2 = pTmp;
      }

      free(result_2);
      *ppInOutData = result_1;
}
```

unbounded until

```
double * dtmc_unbounded_until(const bitset *phi, const bitset *psi) {
    sparse *state_space = get_state_space();
    bitset * EU = get_exist_until(state_space, phi, psi); /* find states not in S_0 */
    bitset * AU = get_always_until(state_space, phi, psi, EU); /* find states in S_1 */
    int i, size = mtx_rows(state_space);
    double * initial = (double *) calloc((size_t) size, sizeof(double)),
        * rhs = (double *) calloc((size_t) size, sizeof(double)), * result = NULL;
    sparse * pP = allocate_sparse_matrix(size, size);
    bitset *dummy = not(AU);
    int *pValidStates=NULL;
    and_result(EU, dummy);
    pValidStates = count_set(dummy);
    /* make ~(Phi /\ ~Psi) states absorbing */
    makeAbsorbingDTMC(state_space, pP, pValidStates);
    i = state_index_NONE;
    while ( (i = get_idx_next_non_zero(AU, i)) != state_index_NONE ) {
            rhs[i] = 1.0;
            initial[i] = 1.0;
    }
    result = unbounded_until_universal(pP, pValidStates, initial, rhs);
    free(rhs); free_bitset(EU); free_bitset(AU); free_bitset(dummy);
    free(pValidStates); free_mtx_wrap(pP);
    return result;
}
```

```
/* Find S0. */
bitset * get_exist_until(const sparse *state_space, const bitset *phi, const bitset *psi)
{
	int i, reach;
	int * states = NULL; /* list of states that are not in S0 */
	bitset * EU; /* bitset of states that are not in S0 */

	copy_bitset(psi, EU);
	states = count_set(psi); /* create list of states */
	reach = states[0]; /* number of states in the list */
	for ( i = 1 ; i <= reach ; i++ ) {
		mtx_walk_column_nodiag_noval(state_space, back_set_j, states[i]) {
			if ( get_bit_val(phi, back_set_j) && ! get_bit_val(EU, back_set_j) ) {
				states = (int *) realloc(states, (reach + 2) * sizeof(int));
				states[++reach] = back_set_j;
				set_bit_val(EU, back_set_j, BIT_ON);
			}
		} end_mtx_walk_column_nodiag_noval;
	}
	free(states); states=NULL;
	return EU; /* result = states that are not in S0 */
}
```

```
double * unbounded_until_universal(sparse * pM, int * pValidStates, double * pX,
        const double * pB)
{
    double err = get_error_bound();
    int max_iterations = get_max_iterations();
    int method;

    /* get (I-P) */
    mult_mtx_cer_const(pM, -1.0, pValidStates);
    add_cer_cons_diagonal_mtx(pM, pValidStates, 1);

    /* solve (I-P)x = i_Psi */
    if( get_method_path() == GJ)
        method = GAUSS_JACOBI;
    else
        method = GAUSS_SEIDEL;

    return solve_initial(method, pM, pX, pB, err, max_iterations, pValidStates);
}
```

Craps as a MRMC input file

Transitions

```
STATES 6
TRANSITIONS 16
1 2 0.22222222222222
1 3 0.11111111111111
1 4 0.16666666666667
1 5 0.22222222222222
1 6 0.27777777777778
2 2 1
3 3 1
4 2 0.08333333333333
4 3 0.16666666666667
4 4 0.75
5 2 0.11111111111111
5 3 0.16666666666667
5 5 0.72222222222222
6 2 0.13888888888889
6 3 0.16666666666667
6 6 0.69444444444444
```

State labels

```
#DECLARATION
win lose
#END
2 win
3 lose
```

```
dnjansen% mrmc dtmc craps.tra craps.lab
 ------------------------------------------------------------
|                    Markov Reward Model Checker             |
|                          MRMC Version 1.5                  |
|                Copyright (C) RWTH Aachen, 2006-2011.       |
|          Copyright (C) The University of Twente, 2004-2008.|
|                              Authors:                      |
|      Ivan S. Zapreev (2004-2011), David N. Jansen (since 2007),|
|         E. Moritz Hahn (2007-2010), Falko Dulat (2009-2010),|
|         Christina Jansen (2007-2008), Sven Johr (2006-2007),|
|          Tim Kemna (2005-2006), Maneesh Khattri (2004-2005)|
|           MRMC is distributed under the GPL conditions     |
|           (GPL stands for GNU General Public License)      |
|          The product comes with ABSOLUTELY NO WARRANTY.    |
|  This is a free software, and you are welcome to redistribute it. |
 ------------------------------------------------------------
 Logic           = PCTL
Loading the 'craps.tra' file, please wait.
States=6, Transitions=16
Loading the 'craps.lab' file, please wait.
The Occupied Space is ??? Bytes.
Type 'help' to get help.
>>P{>0.5} [tt U lose]
Solving the system of linear equations:
    Method: GAUSS-SEIDEL
    Error: 1.000000e-06
    Max iter: 1000000
Gauss Seidel V_B: The number of Gauss-Seidel iterations 3
```

```
 Logic           = PCTL
Loading the 'craps.tra' file, please wait.
States=6, Transitions=16
Loading the 'craps.lab' file, please wait.
The Occupied Space is ??? Bytes.
Type 'help' to get help.
>>P{>0.5} [tt U lose]
Solving the system of linear equations:
    Method: GAUSS-SEIDEL
    Error: 1.000000e-06
    Max iter: 1000000
Gauss Seidel V_B: The number of Gauss-Seidel iterations 3
$ERROR_BOUND: 1.000000e-06
$RESULT: ( 0.5070707, 0.0000000, 1.0000000, 0.6666667, 0.6000000,
0.5454545 )
$STATE: { 1, 3, 4, 5, 6 }

The Total Elapsed Model-Checking Time is 0 milli sec(s).
>>P{>=0.7} [tt U[0,3] win]
$ERROR_BOUND: 1.000000e-06
$RESULT: ( 0.3544239, 1.0000000, 0.0000000, 0.1927083, 0.2493141,
0.3023191 )
$STATE: { 2 }

The Total Elapsed Model-Checking Time is 0 milli sec(s).
>>quit
dnjansen% _
```

```
dnjansen% mrmc dtmc craps10.tra craps10.lab
 ------------------------------------------------------------
|                    Markov Reward Model Checker             |
|                          MRMC Version 1.5                  |
|                Copyright (C) RWTH Aachen, 2006-2011.       |
|          Copyright (C) The University of Twente, 2004-2008.|
|                              Authors:                      |
|      Ivan S. Zapreev (2004-2011), David N. Jansen (since 2007),|
|         E. Moritz Hahn (2007-2010), Falko Dulat (2009-2010),|
|         Christina Jansen (2007-2008), Sven Johr (2006-2007),|
|          Tim Kemna (2005-2006), Maneesh Khattri (2004-2005)|
|           MRMC is distributed under the GPL conditions     |
|           (GPL stands for GNU General Public License)      |
|          The product comes with ABSOLUTELY NO WARRANTY.    |
|  This is a free software, and you are welcome to redistribute it. |
 ------------------------------------------------------------
 Logic           = PCTL
Loading the 'craps10.tra' file, please wait.
States=7, Transitions=21
Loading the 'craps10.lab' file, please wait.
The Occupied Space is ??? Bytes.
Type 'help' to get help.
>>P{<0.3} [!ten U lose]
Solving the system of linear equations:
    Method: GAUSS-SEIDEL
    Error: 1.000000e-06
    Max iter: 1000000
Gauss Seidel V_B: The number of Gauss-Seidel iterations 3
```

```
|         Christina Jansen (2007-2008), Sven Johr (2006-2007),      |
|          Tim Kemna (2005-2006), Maneesh Khattri (2004-2005)       |
|           MRMC is distributed under the GPL conditions            |
|           (GPL stands for GNU General Public License)             |
|          The product comes with ABSOLUTELY NO WARRANTY.           |
|  This is a free software, and you are welcome to redistribute it. |
 -------------------------------------------------------------------
 Logic           = PCTL
Loading the 'craps10.tra' file, please wait.
States=7, Transitions=21
Loading the 'craps10.lab' file, please wait.
The Occupied Space is ??? Bytes.
Type 'help' to get help.
>>P{<0.3} [!ten U lose]
Solving the system of linear equations:
    Method: GAUSS-SEIDEL
    Error: 1.000000e-06
    Max iter: 1000000
Gauss Seidel V_B: The number of Gauss-Seidel iterations 3
$ERROR_BOUND: 1.000000e-06
$RESULT: ( 0.3743895, 0.0000000, 1.0000000, 0.5000000, 0.4615385,
0.4285714, 0.0000000 )
$STATE: { 2, 7 }

The Total Elapsed Model-Checking Time is 0 milli sec(s).
>>quit
dnjansen% _
```