Formal Methods in Software Development

Applications of SMT solving

West University of Timișoara
Faculty of Mathematics and Informatics
Department of Computer Science

Based on slides of the lecture Satisfiability Checking (Erika Ábrahám), RTWH Aachen

WS 2019/2020

Virtual Machine Placement Problem

Assume that we have three virtual machines (VMs) which require 100, 50 and 15 GB hard disk respectively. There are three servers with capabilities 100, 75 and 200 GB in that order. Find out a way to place VMs into servers in order to:

1. Minimize the number of servers used.
2. Minimize the operation cost (the servers have fixed daily costs 10, 5 and 20 USD respectively.)

Virtual Machine Placement Problem

Assume that we have three virtual machines (VMs) which require 100, 50 and 15 GB hard disk respectively. There are three servers with capabilities 100, 75 and 200 GB in that order. Find out a way to place VMs into servers in order to:

1. Minimize the number of servers used.
2. Minimize the operation cost (the servers have fixed daily costs 10, 5 and 20 USD respectively.)

Solution. We first formalize the problem. Then we can translate it into the SMT-LIB format and use an SMT solver to find a solution.

Virtual Machine Placement Problem

Assume that we have three virtual machines (VMs) which require 100, 50 and 15 GB hard disk respectively. There are three servers with capabilities 100, 75 and 200 GB in that order. Find out a way to place VMs into servers in order to:

1. Minimize the number of servers used.
2. Minimize the operation cost (the servers have fixed daily costs 10, 5 and 20 USD respectively.)

Solution. We first formalize the problem. Then we can translate it into the SMT-LIB format and use an SMT solver to find a solution.
Let x_{ij} denote that VM i is placed on the server j and y_j denote that server j is in use. We need to express the followings.

Virtual Machine Placement Problem

Assume that we have three virtual machines (VMs) which require 100, 50 and 15 GB hard disk respectively. There are three servers with capabilities 100, 75 and 200 GB in that order. Find out a way to place VMs into servers in order to:

1. Minimize the number of servers used.
2. Minimize the operation cost (the servers have fixed daily costs 10, 5 and 20 USD respectively.)

Solution. We first formalize the problem. Then we can translate it into the SMT-LIB format and use an SMT solver to find a solution.
Let x_{ij} denote that VM i is placed on the server j and y_j denote that server j is in use. We need to express the followings.

- A VM is on exactly one server: $x_{i1} + x_{i2} + x_{i3} = 1, \quad \forall i \in \{1, ..., 3\}$

Virtual Machine Placement Problem

Assume that we have three virtual machines (VMs) which require 100, 50 and 15 GB hard disk respectively. There are three servers with capabilities 100, 75 and 200 GB in that order. Find out a way to place VMs into servers in order to:

1. Minimize the number of servers used.
2. Minimize the operation cost (the servers have fixed daily costs 10, 5 and 20 USD respectively.)

Solution. We first formalize the problem. Then we can translate it into the SMT-LIB format and use an SMT solver to find a solution.

Let x_{ij} denote that VM i is placed on the server j and y_j denote that server j is in use. We need to express the followings.

- A VM is on exactly one server: $x_{i1} + x_{i2} + x_{i3} = 1, \quad \forall i \in \{1, ..., 3\}$
- A used server has at least a VM on it: $(x_{1j}{=}1) \vee ... \vee (x_{3j}{=}1) \Rightarrow (y_j{=}1), j{=}\overline{1,3}$

Virtual Machine Placement Problem

Assume that we have three virtual machines (VMs) which require 100, 50 and 15 GB hard disk respectively. There are three servers with capabilities 100, 75 and 200 GB in that order. Find out a way to place VMs into servers in order to:

1. Minimize the number of servers used.
2. Minimize the operation cost (the servers have fixed daily costs 10, 5 and 20 USD respectively.)

Solution. We first formalize the problem. Then we can translate it into the SMT-LIB format and use an SMT solver to find a solution.
Let x_{ij} denote that VM i is placed on the server j and y_j denote that server j is in use. We need to express the followings.

- A VM is on exactly one server: $x_{i1} + x_{i2} + x_{i3} = 1, \quad \forall i \in \{1, ..., 3\}$
- A used server has at least a VM on it: $(x_{1j}{=}1) \vee ... \vee (x_{3j}{=}1) \Rightarrow (y_j{=}1), j{=}\overline{1,3}$
- Capability constraints: $100x_{11} + 50x_{21} + 15x_{31} \leq 100y_1$
  $100x_{12} + 50x_{22} + 15x_{32} \leq 75y_2$
  $100x_{13} + 50x_{23} + 15x_{33} \leq 200y_3$

Virtual Machine Placement Problem (cont'd)

There are various ways of encoding the variables of the problem, for example:

(Variant 1) as integers

(Variant 2) as real

(Variant 3) as bool

(Variant 4) using `assert-soft` constraints

Virtual Machine Placement Problem (cont'd)

There are various ways of encoding the variables of the problem, for example:

(Variant 1) as integers

(Variant 2) as real

(Variant 3) as bool

(Variant 4) using `assert-soft` constraints

Variant 1.

Virtual Machine Placement Problem (cont'd)

There are various ways of encoding the variables of the problem, for example:

(Variant 1) as integers
(Variant 2) as real
(Variant 3) as bool
(Variant 4) using `assert-soft` constraints

Variant 1. We declare each variable as integer, e.g.:

```
(declare-const x11 Int)
```

We also need to ensure that variables are 0/1, e.g.:

```
(assert (and (>= x11 0) (>= x12 0) (>= x13 0) (>= x21 0)...
(assert (and (<= y1 1) (<= y2 1) (<= y3 1)))
```

Another variant for encoding the 0/1 integers is:

```
(assert (or (>= x11 0) (<= x11 1)
(assert (or (>= x12 0) (<= x12 1)...
```

Virtual Machine Placement Problem (cont'd)

There are various ways of encoding the variables of the problem, for example:

(Variant 1) as integers

(Variant 2) as real

(Variant 3) as bool

(Variant 4) using `assert-soft` constraints

Variant 1. Constraint of type 2 can be encoded in 2 ways. For example:

```
(assert (and (>= y1 x11) (>= y1 x21) (>= y1 x31)))
...
```

Or as:

```
(assert (implies (= y1 1) (or (= x11 1) (= x21 1) (= x31 1)
...
```

Virtual Machine Placement Problem (cont'd)

There are various ways of encoding the variables of the problem, for example:

(Variant 1) as integers

(Variant 2) as real

(Variant 3) as bool

(Variant 4) using `assert-soft` constraints

Variant 2. We declare each variable as real. The constraints should be the same as for the integer encoding.

Virtual Machine Placement Problem (cont'd)

There are various ways of encoding the variables of the problem, for example:

(Variant 1) as integers

(Variant 2) as real

(Variant 3) as bool

(Variant 4) using `assert-soft` constraints

Variant 3. We declare each variable as bool. The capability constraints require only integer/real variables, so we need to transform the bool variables into integer/real. This can be done by declaring a function as follows: `(define-fun bool_to_int ((b Bool)) Int (ite b 1 0) )` and cast the bool variables to int/real, e.g.

```
(assert (<= (+ (* 100 (bool_to_int x11)) ...  )  (...)))
```

Virtual Machine Placement Problem (cont'd)

There are various ways of encoding the variables of the problem, for example:

(Variant 1) as integers
(Variant 2) as real
(Variant 3) as bool
(Variant 4) using `assert-soft` constraints

Variant 4. Use `assert-soft` constraints (soft constraints) (see `https://rise4fun.com/Z3/tutorialcontent/optimization`). For our problem, soft constraints can be used to encode the optimization goals:

```
(assert-soft (not y1) :id num_servers) ...
(assert-soft (not y1) :id costs :weight 10) ...
```

The `assert-soft` command represents MaxSMT (maximize the number of constraints which can be satisfied) which tries to maximize the weighted sum of boolean expressions belonged to the same `id`. Since we are doing minimization, negation is needed to take advantage of MaxSMT support.

Job shop scheduling Problem

Job shop scheduling is an optimization problem in computer science and operations research in which jobs are assigned to resources at particular times. The most basic version is as follows. We are given n jobs J_1, J_2, ..., J_n of varying processing times, which need to be scheduled on m machines with varying processing power, while trying to minimize the *makespan*. The *makespan* is the total length of the schedule (that is, when all the jobs have finished processing). Additionally, the following constraints might be involved:

- *Precedence.* Between two jobs which want to take a machine.
- *Resource.* Machines execute at most one job at a time.

Job shop scheduling Problem (cont'd)

Example of a Job shop scheduling problem

$d_{i,j}$	Machine 1	Machine 2
Job 1	2	1
Job 2	3	1
Job 3	2	3

max = 8

Let d_{ij} be the duration of Job i on Machine j. For example $d_{11} = 2$ time units (TU), $d_{12} = 1$ TU, etc. The maximal makespan is 8 TU.
The problem can be formalized as follows. We consider the variable t_{ij} representing "time required for job i on machine j." ($i = \overline{1,3}$, $j = \overline{1,2}$)

- *precedence*. For example, for Job 1 we have:

$$(t_{1,1} \geq 0) \wedge (t_{1,2} \geq t_{1,1} + 2) \wedge (t_{1,2} + 1 \leq 8)$$

- *resource*. For example, we have "Job 1 on machine 1 is scheduled either before or after job 2 on the same machine":

$$(t_{1,1} \geq t_{2,1} + 3) \vee (t_{2,1} \geq t_{1,1} + 2)$$

Job shop scheduling Problem (cont'd)

- Using Z3 SMT solver, we can find a suitable schedule, for example $t_{31} = 2,\ t_{21} = 4,\ t_{22} = 7,\ t_{32} = 4,\ t_{12} = 2,\ t_{11} = 0.$

Job shop scheduling Problem (cont'd)

- Using Z3 SMT solver, we can find a suitable schedule, for example $t_{31} = 2,\ t_{21} = 4,\ t_{22} = 7,\ t_{32} = 4,\ t_{12} = 2,\ t_{11} = 0.$
- Is it optimal?

Job shop scheduling Problem (cont'd)

- Using Z3 SMT solver, we can find a suitable schedule, for example $t_{31} = 2$, $t_{21} = 4$, $t_{22} = 7$, $t_{32} = 4$, $t_{12} = 2$, $t_{11} = 0$.
- Is it optimal?
- It seems like it is not optimal since we have unused times on VM2. For optimality add the constraints which minimizes the maximum finish times on each machine:

```
(define-fun max ((x Int) (y Int)) Int (ite (< x y) y x))
(minimize (max (+ t11 2) (max (+ t21 3) (+ t31 2))))
(minimize (max (+ t12 1) (max (+ t22 1) (+ t32 3))))
```