NetSAT: Automated reasoning methods for verification and configuration of computer networks

Marco Gario

EMCL / NICTA

February 18, 2011

Content

Outline

1. Overview
   - Background info
2. Roadmap
   - Formalization
   - As Planning
   - As SAT
   - Reconfiguration
   - Complexity Results
3. Conclusions
   - Open Issues
   - Questions, Critics, Suggestions?

Background info

- NICTA Canberra Research Lab
- 10 weeks ($\approx$2 months)
- Supervisors: Jussi Rintanen, Alban Grastien.

  Leading the *Smart Grid* project

Background info

- NICTA Canberra Research Lab
- 10 weeks ($\approx$2 months)
- Supervisors: Jussi Rintanen, Alban Grastien.
  Leading the *Smart Grid* project

Rough idea of the problem

Given a computer network of which we know the configuration and the intended behaviour (*policy*), we want to check whether the current configuration "satisfies" the policy or not; if not we want to know what are the alternative configurations that can satisfy it (if any).

Eg.:

- Only the students inside the university network should be able to access the eBooks,
- Can the user A reach the service B?

Managing configurations in *complex* environments is not trivial.

Rough idea of the problem

Given a computer network of which we know the configuration and the intended behaviour (*policy*), we want to check whether the current configuration “satisfies” the policy or not; if not we want to know what are the alternative configurations that can satisfy it (if any).

Eg.:

- Only the students inside the university network should be able to access the eBooks,
- Can the user A reach the service B?

Managing configurations in *complex* environments is not trivial.

Rough idea of the problem

Given a computer network of which we know the configuration and the intended behaviour (*policy*), we want to check whether the current configuration "satisfies" the policy or not; if not we want to know what are the alternative configurations that can satisfy it (if any).

Eg.:

- Only the students inside the university network should be able to access the eBooks,
- Can the user A reach the service B?

Managing configurations in *complex* environments is not trivial.

Rough idea of the problem

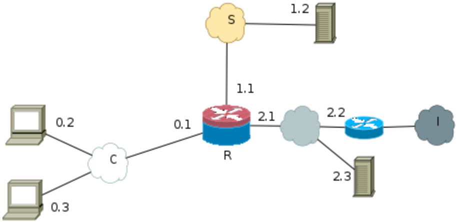
Given a computer network of which we know the configuration and the intended behaviour (*policy*), we want to check whether the current configuration "satisfies" the policy or not; if not we want to know what are the alternative configurations that can satisfy it (if any).

Eg.:

- Only the students inside the university network should be able to access the eBooks,
- Can the user A reach the service B?

Managing configurations in *complex* environments is not trivial.

Example Network



Related Work

- configAssure [1] (2008):
  - Alloy modelling language → KodKod: [2] a constraint solver for relational logic
  - Complexity in the specification of the requirements as Datalog
  - KodKod solves the problem by reducing to SAT
  - Commercial product *IPAssure* by Telecordia
- ConfigChecker [3] (2009):
  - More similar to this work
  - Extension of CTL to specify requirements
  - BDD based
  - Many modelling problem (as directionality) are not explicit in the reports

Outline

1. Overview
   - Background info
2. Roadmap
   - Formalization
   - As Planning
   - As SAT
   - Reconfiguration
   - Complexity Results
3. Conclusions
   - Open Issues
   - Questions, Critics, Suggestions?

Formalization of the problem

- What level of detail do we want?
- How can we characterize network components?
- How does the policy looks like?

Detail Level

We need to decide how much into detail we want to go. We consider only the TCP and IP level of the TCP/IP suite. Therefore we talk mainly about *packets* of data.
We consider the following components:

- Host,
- Router,
- Firewall,
- NAT

Detail Level

We need to decide how much into detail we want to go. We consider only the TCP and IP level of the TCP/IP suite. Therefore we talk mainly about *packets* of data.

We consider the following components:

- Host, that has an IP Address, can be Source or Destination of a connection
- Router,
- Firewall,
- NAT

Detail Level

We need to decide how much into detail we want to go. We consider only the TCP and IP level of the TCP/IP suite. Therefore we talk mainly about *packets* of data.
We consider the following components:

- Host,
- Router, that is connected to multiple "networks" and can decide to which one to forward an incoming packet
- Firewall,
- NAT

Detail Level

We need to decide how much into detail we want to go. We consider only the TCP and IP level of the TCP/IP suite. Therefore we talk mainly about *packets* of data.
We consider the following components:

- Host,
- Router,
- Firewall, that can allow or deny the transit of a packet
- NAT

Detail Level

We need to decide how much into detail we want to go. We consider only the TCP and IP level of the TCP/IP suite. Therefore we talk mainly about *packets* of data.

We consider the following components:

- Host,
- Router,
- Firewall,
- NAT that can modify the content of a packet

Rules

All network components are rule-based:

	Condition	Action
Routing	Destination IP	Next Hop
Firewall	Any TCP/IP field	Accept, Deny
NAT	Any TCP/IP field	Modify any TCP/IP field

Assumption: Rules are *deterministic* and are *independent* one from the other.

This structure (Condition,Action) can be used to describe the behaviour of many components in networking (eg. IPSec).

When dealing with reconfiguration we allow only the modification of the rules: we exclude, eg., topological modifications.

Rules

All network components are rule-based:

	Condition	Action
Routing	Destination IP	Next Hop
Firewall	Any TCP/IP field	Accept, Deny
NAT	Any TCP/IP field	Modify any TCP/IP field

Assumption: Rules are *deterministic* and are *independent* one from the other.
This structure (Condition,Action) can be used to describe the behaviour of many components in networking (eg. IPSec).
When dealing with reconfiguration we allow only the modification of the rules: we exclude, eg., topological modifications.

Rules

All network components are rule-based:

	Condition	Action
Routing	Destination IP	Next Hop
Firewall	Any TCP/IP field	Accept, Deny
NAT	Any TCP/IP field	Modify any TCP/IP field

Assumption: Rules are *deterministic* and are *independent* one from the other.

This structure (Condition,Action) can be used to describe the behaviour of many components in networking (eg. IPSec).

When dealing with reconfiguration we allow only the modification of the rules: we exclude, eg., topological modifications.

Network property

We define the following decision problem:

Basic Reachability Problem

Given:

- a network configuration $\mathcal{C}$
- an initial position Pos_0,
- a formula characterising a non-empty set of initial packets τ,
- a formula characterising the path *VALID*
- a final position Pos_G,
- and an integer n

Is it possible in the network $\mathcal{C}$ for all the packets p (s.t. $p \models \tau$) starting from Pos_0 to reach Pos_G in n steps (or less) satisfying the condition *VALID*?

Policy

We define also the Unreachability:

In the network $\mathcal{C}$ **no one** of the packets p (s.t. $p \models \tau$) starting from Pos_0 will reach Pos_G in n steps (or less) satisfying the condition *VALID*

A *Policy* is a collection of Network Properties (Reachability and Unreachability)

A *Policy* holds *iff* all the properties hold

Outline

1 Overview
- Background info

2 Roadmap
- Formalization
- As Planning
- As SAT
- Reconfiguration
- Complexity Results

3 Conclusions
- Open Issues
- Questions, Critics, Suggestions?

Planning encoding

First solution as a Planning problem

- Intuitively we model each component as a set of actions performed on a single packet
- Is there a path/plan from A to B?
- We generated PDDL files and solved some toy example.
- Not “suitable” if we want to verify that there's **no** path from A to B: we need a *complete* planner.

Planning encoding

First solution as a Planning problem

- Intuitively we model each component as a set of actions performed on a single packet
- Is there a path/plan from A to B?
- We generated PDDL files and solved some toy example.
- Not "suitable" if we want to verify that there's **no** path from A to B: we need a *complete* planner.

Outline

1. Overview
   - Background info
2. Roadmap
   - Formalization
   - As Planning
   - As SAT
   - Reconfiguration
   - Complexity Results
3. Conclusions
   - Open Issues
   - Questions, Critics, Suggestions?

SAT encoding

SAT reformulation of the verification problem:

- Planning as SAT
- SAT approach gives us a bit more flexibility
- It turns out to be very similar to Bounded Model Checking!
- We **do** have a bound!
- We have a complete method for verification

SAT encoding

SAT reformulation of the verification problem:

- Planning as SAT
- SAT approach gives us a bit more flexibility
- It turns out to be very similar to Bounded Model Checking!
- We **do** have a bound!
- We have a complete method for verification

SAT encoding

SAT reformulation of the verification problem:

- Planning as SAT
- SAT approach gives us a bit more flexibility
- It turns out to be very similar to Bounded Model Checking!
- We **do** have a bound!
- We have a complete method for verification

SAT encoding

SAT reformulation of the verification problem:

- Planning as SAT
- SAT approach gives us a bit more flexibility
- It turns out to be very similar to Bounded Model Checking!
- We **do** have a bound!
- We have a complete method for verification

Outline

1. Overview
   - Background info
2. Roadmap
   - Formalization
   - As Planning
   - As SAT
   - Reconfiguration
   - Complexity Results
3. Conclusions
   - Open Issues
   - Questions, Critics, Suggestions?

Reconfiguration

We want to find a network $\mathcal{C}$ that satisfies the policy:

- Even preserving the topology, we still have an huge search space (eg. $\approx 2^{200}$ possible configurations for *each* network element)
- We simplify the problem assuming that we are given a set of *available* configurations.
- Encoding of the reconfiguration problem as 2QBF:
  - $\exists\forall$: $\exists$ configuration $\forall$ packets
  - Not many solvers are available. The ones tested couldn't even solve the problem with a single configuration.
- We manage to solve this problem “faster” by using an incremental SAT solver!

Reconfiguration

We want to find a network $\mathcal{C}$ that satisfies the policy:

- Even preserving the topology, we still have an huge search space (eg. $\approx 2^{200}$ possible configurations for *each* network element)
- We simplify the problem assuming that we are given a set of *available* configurations.
- Encoding of the reconfiguration problem as 2QBF:
  - $\exists\forall$: $\exists$ configuration $\forall$ packets
  - Not many solvers are available. The ones tested couldn't even solve the problem with a single configuration.
- We manage to solve this problem "faster" by using an incremental SAT solver!

Reconfiguration

We want to find a network $\mathcal{C}$ that satisfies the policy:

- Even preserving the topology, we still have an huge search space (eg. $\approx 2^{200}$ possible configurations for *each* network element)
- We simplify the problem assuming that we are given a set of *available* configurations.
- Encoding of the reconfiguration problem as 2QBF:
  - $\exists\forall$: $\exists$ configuration $\forall$ packets
  - Not many solvers are available. The ones tested couldn't even solve the problem with a single configuration.
- We manage to solve this problem "faster" by using an incremental SAT solver!

Reconfiguration

We want to find a network $\mathcal{C}$ that satisfies the policy:

- Even preserving the topology, we still have an huge search space (eg. $\approx 2^{200}$ possible configurations for *each* network element)
- We simplify the problem assuming that we are given a set of *available* configurations.
- Encoding of the reconfiguration problem as 2QBF:
  - $\exists\forall$: $\exists$ configuration $\forall$ packets
  - Not many solvers are available. The ones tested couldn't even solve the problem with a single configuration.
- We manage to solve this problem "faster" by using an incremental SAT solver!

Outline

1. Overview
   - Background info
2. Roadmap
   - Formalization
   - As Planning
   - As SAT
   - Reconfiguration
   - Complexity Results
3. Conclusions
   - Open Issues
   - Questions, Critics, Suggestions?

Complexity

We study two types of complexity:

- Packet
- Network

	Network	Packet
BRP	P	coNP-complete
BRP*	P	coNP-complete
Reconf	NP-complete	Σ_p^2 (?)

Complexity

We study two types of complexity:

- Packet
- Network

	Network	Packet
BRP	P	coNP-complete
BRP*	P	coNP-complete
Reconf	NP-complete	Σ_p^2 (?)

Outline

1. Overview
   - Background info
2. Roadmap
   - Formalization
   - As Planning
   - As SAT
   - Reconfiguration
   - Complexity Results
3. Conclusions
   - Open Issues
   - Questions, Critics, Suggestions?

Open Issues

Many interesting open issues:

- How to generate the configuration set $\mathcal{C}$?
- Scaling Tests
- Disjointness of rules
- Higher levels with more complex interactions (eg. TCP sessions)
- Rewriting as LTL SAT problem

Outline

How did I spent my time

- 50% Researching and reading papers (of which most of the time was spent actually looking for them)
- 30% Optimising and developing
- 15% Studying/Developing theoretical aspects (mainly through written mini-reports)
- 5% Disturbing my supervisors!

Comments

- Questions?
- Critics?
- Suggestions?

References

- Narain, S. and Levin, G. and Malik, S. and Kaul, V., Declarative infrastructure configuration synthesis and debugging, Journal of Network and Systems Management, 2008
- http://alloy.mit.edu/kodkod/
- Ehab Al-Shaer, Will Marrero, Adel El-Atawy and Khalid Elbadawi, Network Security Configuration in A Box: End-to-End Security
- Goultiaeva, Alexandra and Bacchus, Fahiem, Exploiting Circuit Representations in QBF Solving, LNCS-6175 Theory and Applications of Satisfiability Testing - SAT 2010, 2010.
- P Manolios, S Srinivasan, D Vroon, BAT: The bit-level analysis tool, Computer Aided Verification, 2007 - Springer