From Boolean to Quantitative System Specifications

Tom Henzinger
EPFL
1 The Quantitative Agenda
2 Some Basic Open Problems
3 Some Promising Directions
The Boolean Agenda

Program/System  Property/Specification

Analysis

Yes/No
The Boolean Agenda

Program/System Property/Specification

Analysis

Yes/No
- perhaps a proof
- perhaps some counterexamples
- perhaps even a proposed fix
The Boolean Agenda

Structure

Program/System

Property/Specification

Formula

Satisfaction Relation

Yes/No
- perhaps a proof
- perhaps some counterexamples
- perhaps even a proposed fix
The Boolean Agenda

Transition system.

Program/System  Property/Specification

Every request is followed by a grant.

Analysis

Yes/No

- perhaps a proof
- perhaps some counterexamples
- perhaps even a proposed fix
Every request is followed by a grant within 5 time units.

- perhaps a proof
- perhaps some counterexamples
- perhaps even a proposed fix
The Boolean Agenda

Markov process.

Quantitative Program/System

Quantitative Property/Specification

Analysis

Yes/No
- perhaps a proof
- perhaps some counterexamples
- perhaps even a proposed fix

Every request is followed by a grant within probability $\frac{1}{2}$. 
The Boolean Agenda

Markov process.

Quantitative Program/System

Quantitative Property/Specification

Every request is followed by a grant within probability 1/2.

Analysis

- perhaps a proof
- perhaps some counterexamples
- perhaps even a proposed fix
The Quantitative Agenda

Quantitative Program/System \( \rightarrow \) Analysis \( \rightarrow \) Quantitative Property/Specification

\( R \)

-measure of “fit” between system and spec
-could be cost, quality, etc.
The Quantitative Agenda

Every request is followed by a grant.

The less time between requests and grants, the better.

- measure of “fit” between system and spec
  - could be cost, quality, etc.
Every request is followed by a grant.

- measure of “fit” between system and spec
- could be cost, quality, etc.

The fewer unnecessary grants, the better.
The Quantitative Agenda

Q1 Assigning values to behaviors
   Boolean case: correct vs. incorrect behaviors

Q2 Assigning values to systems/properties
   Boolean case: sets of behaviors (nondeterminism)

Q3 Assigning values to pairs of systems
   Boolean case: preorders on systems
Q1 Assigning Values To Behaviors

a. Probabilities

b. Resource use
   - worst case vs. average case (e.g. response time, QoS)
   - peak vs. accumulative (e.g. power consumption)

c. Quality measures
   - discounting vs. long-run averaging (e.g. reliability)
Q1 Assigning Values To Behaviors

a: ok
b: fail

Discounted value $(0 < d < 1)$:

- $aaaaaaa... \quad 1$
- $aaaaaaab... \quad 1 - d^8$
- $aab... \quad 1 - d^3$
- $b... \quad 0$

Long-run average value:

- $aaaaaaaa... \quad 1$
- $aaabaaabaaab... \quad 1 - \frac{1}{4}$
- $abaabaaab... \quad 0$
Assigning Values To Systems

x: behaviors
w: observations
A,B: systems

\[ A(w) = \sup_x \{ \text{val}(x) : \text{obs}(x) = w \} \]

\[ B(w) = \exp_x \{ \text{val}(x) : \text{obs}(x) = w \} \]
Q2, Q3 Assigning Values To Systems

x: behaviors
w: observations
A,B: systems

\[ A(w) = \sup_x \{ \text{val}(x) : \text{obs}(x) = w \} \]

\[ B(w) = \exp_x \{ \text{val}(x) : \text{obs}(x) = w \} \]

\[ \text{diff}(A,B) = \sup_w \{ |A(w) - B(w)| \} \]

Compositionality: \( \text{diff}(A || B, A' || B) \leq f(\text{diff}(A, A')) \) [AFHMS].
Is there a Quantitative Framework with
-an appealing mathematical formulation,
-useful expressive power, and
-good algorithmic properties?

(Like the boolean theory of $\omega$-regularity.)
Outline

1 The Quantitative Agenda
2 Some Basic Open Problems
3 Some Promising Directions
Property = Language

Alphabet: \( \Sigma \)
\( \Sigma = \{a, b, c\} \)

Language: \( L \subseteq \Sigma^\omega \)
\( L = (a^+b^+)(a^\omega \cup c^\omega) \cup (a^b)^\omega \)

abaabaaabcccccc... \( \in L \)
abcabc... \( \notin L \)
Boolean Language

Alphabet:
\[ \Sigma \]
\[ \Sigma = \{a, b, c\} \]

Language:
\[ L \subseteq \Sigma^\omega \]
\[ L = (a^*b)^+(a^\omega \cup c^\omega) \cup (a^*b)^\omega \]

\[ abaabaaabcccccc... \in L \]
\[ abcabc... \notin L \]

\[ L : \Sigma^\omega \rightarrow \mathbb{B} \]
Specification = Automaton

\[ Q \]
\[ \lambda: Q \rightarrow \Sigma \]
\[ q_0 \in Q \]
\[ \Gamma \]
\[ \delta: Q \times \Gamma \rightarrow Q \]

states
labeling
initial state
choices
transition function

\[ \Gamma = \{0, 1\} \]
Specification = Automaton

Q
λ: Q → Σ
q_0 ∈ Q
Γ
δ: Q × Γ → Q

states
labeling
initial state
choices
transition function

Γ = \{0,1\}
L(A) = (a+b)^+((a^ω∪c^ω)∪(a+b)^ω)
Automaton

$Q$ states

$\lambda: Q \rightarrow \Sigma$ labeling

$q_0 \in Q$ initial state

$\Gamma$ choices

$\delta: Q \times \Gamma \rightarrow Q$ transition function

A: 0 1 0 1 0, 1

"scheduler" 0101111... $\rightarrow$ aababccc... "outcome"
Automaton

$Q$  states
$\lambda : Q \rightarrow \Sigma$  labeling
$q_0 \in Q$  initial state
$\Gamma$  choices
$\delta : Q \times \Gamma \rightarrow Q$  transition function

Scheduler:  $x : Q^+ \rightarrow \Gamma$
S ... set of schedulers

Outcome:  $f(x) = q_0q_1q_2 ...$
where $\forall i : q_{i+1} = \delta(q_i, x(q_0...q_i))$

Language:  $L = \{ \lambda(f(x)) : x \in S \}$
Automaton

\[ Q \]
\[ \lambda : Q \rightarrow \Sigma \]
\[ q_0 \in Q \]
\[ \Gamma \]
\[ \delta : Q \times \Gamma \rightarrow Q \]

states
labeling
initial state
choices
transition function

Scheduler: \[ x : Q^+ \rightarrow \Gamma \]
S ... set of schedulers

Outcome: \[ f(x) = q_0 q_1 q_2 \ldots \]
where \[ \forall i : q_{i+1} = \delta(q_i, x(q_0 \ldots q_i)) \]

Language: \[ L = \{ \lambda(f(x)) : x \in S \} \]
\[ L(w) = \sup\{ f(x) : x \in S \text{ s.t. } \lambda(f(x)) = w \} \]
Language Inclusion

Given two automata $A$ and $B$, is $L(A) \subseteq L(B)$?

i.e. $\forall w \in \Sigma^\omega : L(A)(w) \leq L(B)(w)$
Satisfaction = Language Inclusion

Given two automata A and B, is \( L(A) \subseteq L(B) \)?

i.e. \( \forall w \in \Sigma^\omega : L(A)(w) \leq L(B)(w) \)

For finite automata, PSPACE-complete.
Probabilistic Language

Word: element of $\Sigma^\omega$
Probabilistic Word: probability space on $\Sigma^\omega$
Probabilistic Language: set of probabilistic words

\[ w: \begin{align*}
ab\Sigma^\omega &\to 1/2 \\
aab\Sigma^\omega &\to 1/4 \\
aaab\Sigma^\omega &\to 1/8 \\
&\text{...}
\end{align*} \]
Markov Decision Process

- $Q$: states
- $\lambda: Q \to \Sigma$: labeling
- $q_0 \in Q$: initial state
- $\Gamma$: choices
- $\delta: Q \times \Gamma \to D(Q)$: transition function

A: 

- From state a:
  - To state a: 0: 0.5
  - To state b: 0: 0.5, 1: 1

- From state b:
  - To state a: 0: 0.5, 1: 1
  - To state c: 0: 0.5

- From state c:
  - To state c: 0, 1

B: 

- From state c:
  - To state b: 0: 0.5, 1: 1
Markov Decision Process

\[ Q \]
\[ \lambda : Q \rightarrow \Sigma \] states labeling
\[ q_0 \in Q \] initial state
\[ \Gamma \] choices
\[ \delta : Q \times \Gamma \rightarrow D(Q) \] transition function

\[ 0: 0.5 \quad 0: 0.5 \quad 0: 0.5 \quad 0: 0.5 \]
\[ a \quad b \quad c \]

\[ 0101111... \rightarrow abccc... \rightarrow 1/2 \]
\[ aabccc... \rightarrow 1/4 \]
\[ ... \]
Markov Decision Process

\[ Q \quad \lambda: Q \to \Sigma \quad \text{states} \]
\[ q_0 \in Q \quad \text{labeling} \]
\[ \Gamma \quad \text{initial state} \]
\[ \delta: Q \times \Gamma \to D(Q) \quad \text{choices} \]
\[ \text{transition function} \]

Pure scheduler: \( x: Q^+ \to \Gamma \)
Probabilistic scheduler: \( x: Q^+ \to D(\Gamma) \)
Markov Decision Process

\[
\begin{align*}
Q & \quad \text{states} \\
\lambda : Q & \to \Sigma \quad \text{labeling} \\
q_0 & \in Q \quad \text{initial state} \\
\Gamma & \quad \text{choices} \\
\delta : Q \times \Gamma & \to D(Q) \quad \text{transition function}
\end{align*}
\]
Probabilistic Language Inclusion

Given two MDPs $A$ and $B$, is $L(A) \subseteq L(B)$?
Probabilistic Language Inclusion

Given two MDPs A and B, is $L(A) \subseteq L(B)$?
Given two MDPs A and B, is $L(A) \subseteq L(B)$?

Open even if specification B is deterministic (i.e. $|\Gamma| = 1$) and implementation scheduler required to be pure. If both sides are deterministic, then it can be solved in polynomial time (equivalence of Rabin’s probabilistic automata) [Tzeng, DHR].
Quantitative Language

Language: \[ L: \Sigma^\omega \rightarrow \mathbb{B} \]

Quantitative Language: \[ L: \Sigma^\omega \rightarrow \mathbb{R} \]

\[
L(ab^\omega) = \frac{1}{2} \\
L(aab^\omega) = \frac{1}{4} \\
L(aaab^\omega) = \frac{1}{8} \\
\ldots
\]
Weighted Automaton

\[ Q \]
\[ \lambda: Q \to \Sigma \]
\[ q_0 \in Q \]
\[ \Gamma \]
\[ \delta: Q \times \Gamma \to \mathbb{R} \times Q \]

states
labeling
initial state
choices
transition function

\[
A:
\begin{align*}
\text{a} & \xrightarrow{0; 4} \\
\text{b} & \xrightarrow{1; 2} \xrightarrow{1; 1} \text{c} \\
\text{a} & \xrightarrow{0; 0} \\
\text{b} & \xrightarrow{1; 1} \text{c}
\end{align*}
\]
Weighted Automaton

$Q$ states
$\lambda: Q \rightarrow \Sigma$ labeling
$q_0 \in Q$ initial state
$\Gamma$ choices
$\delta: Q \times \Gamma \rightarrow \mathbb{R} \times Q$ transition function

A:

Max value:

0101111... $\rightarrow$ aababccc...; 4
1111111... $\rightarrow$ abccc...; 2
Weighted Automaton

\[ Q \]
\[ \lambda: Q \rightarrow \Sigma \] states
\[ q_0 \in Q \] labeling
\[ \Gamma \] initial state
\[ \delta: Q \times \Gamma \rightarrow \mathbb{R} \times Q \] choices
\[ \text{transition function} \]

Outcome: \[ f(x) = q_0 v_1 q_1 v_2 q_2 \ldots \]
where \[ \forall \, i : (v_{i+1}, q_{i+1}) = \delta(q_i, x(q_0 \ldots q_i)) \]

Max value: \[ \text{val}(q_0 v_1 q_1 v_2 q_2 \ldots) = \sup \{ v_i : i \geq 1 \} \]
Weighted Automaton

- $Q$: states
- $\lambda: Q \rightarrow \Sigma$: labeling
- $q_0 \in Q$: initial state
- $\Gamma$: choices
- $\delta: Q \times \Gamma \rightarrow \mathbb{R} \times Q$: transition function

**Outcome:**

$f(x) = q_0v_1q_1v_2q_2...$

where $\forall \ i : (v_{i+1}, q_{i+1}) = \delta(q_i, x(q_0...q_i))$

**Max value:**

$\text{val}(q_0v_1q_1v_2q_2...) = \sup \{ v_i : i \geq 1 \}$

**q-Language:**

$L(w) = \sup \{ \text{val}(f(x)) : x \in S \text{ s.t. } \lambda(f(x)) = w \}$
Max value: \[ \text{val}(q_0v_1q_1v_2q_2...) = \sup \{ v_i : i \geq 1 \} \]
(only 0 and 1 costs: finite automaton)

Limsup value: \[ \text{val} = \lim_{n \to \infty} \sup \{ v_i : i \geq n \} \]
(only 0 and 1 costs: Buechi automaton)
Different Value Functions

Max value: \( \text{val}(q_0v_1q_1v_2q_2...) = \sup \{ v_i : i \geq 1 \} \)
(only 0 and 1 costs: finite automaton)

Limsup value: \( \text{val} = \lim_{n \to \infty} \sup \{ v_i : i \geq n \} \)
(only 0 and 1 costs: Buechi automaton)

Limavg value: \( \text{val} = \lim_{n \to \infty} \frac{1}{n} \cdot \sum_{1 \leq i \leq n} v_i \)
Different Value Functions

Max value: \( \text{val}(q_0v_1q_1v_2q_2...)=\sup\{v_i : i \geq 1\} \)
(only 0 and 1 costs: finite automaton)

Limsup value: \( \text{val}=\lim_{n \to \infty} \sup\{v_i : i \geq n\} \)
(only 0 and 1 costs: Buechi automaton)

Limavg value: \( \text{val}=\lim_{n \to \infty} \frac{1}{n} \cdot \sum_{1 \leq i \leq n} v_i \)

Discounted: \( \text{val}=\sum_{i \geq 1} d^i \cdot v_i \) for some \( 0 < d < 1 \)
Weighted Automaton

Limsup value:
- $01010101 \ldots \rightarrow \text{aabababab}\ldots; 2$
- $11111111 \ldots \rightarrow \text{abccc}\ldots; 0$

Limavg value:
- $01010101 \ldots \rightarrow \text{aabababab}\ldots; 1$
- $11111111 \ldots \rightarrow \text{abccc}\ldots; 0$

Discounted:
- $01010101 \ldots \rightarrow \text{aabababab}\ldots; 2.66\ldots$
- $11111111 \ldots \rightarrow \text{abccc}\ldots; 1.25$

(d = 0.5)
Given two cost automata $A$ and $B$, is 
$\forall w \in \Sigma^\omega : L(A)(w) \leq L(B)(w)$?
Quantitative Language Inclusion

Given two cost automata $A$ and $B$, is

$$\forall w \in \Sigma^\omega : L(A)(w) \leq L(B)(w)$$

For max and limsup values: PSPACE. For limavg and discounted values: Open.
Given two cost automata $A$ and $B$, is
$\forall w \in \Sigma^\omega : L(A)(w) \leq L(B)(w)$ ?

For max and limsup values: PSPACE. For limavg and discounted values: Open.

If specification $B$ is deterministic, then it can be solved in polynomial time [CDH].
Expressiveness [CDH]

E.g. LimAvg automata not determinizable:

\[ \Sigma^* b^\omega \] expressible by nondeterministic LimAvg automaton.

\[ \Sigma^* b^\omega \] not expressible by deterministic limAvg automaton.

Every b-cycle would need weight 1.
Consider \( w_n = (ab^n)^\omega \).
Then \( \text{val}(w_n) = 1 \) for sufficiently large \( n \), but \( w_n \not\in \Sigma^* b^\omega \).
Outline

1 The Quantitative Agenda
2 Some Basic Open Problems
3 Some Promising Directions
The Boolean Agenda

System Specification

Analysis

Yes/No
The Boolean Agenda

Specification $\omega$-regular Automaton

Synthesis

Correct System
3.1 Quantitative Synthesis

Quantitative Specification \(\rightarrow\) Synthesis \(\rightarrow\) Optimal System

[Jobstmann et al. THURSDAY]
Following a request, all steps until the next grant are penalized.
Following a request, all repeated grants are penalized.
3.2 Robust Systems

A Robustness as Mathematical Continuity:

- small input changes cause small output changes
- only possible in a quantitative framework

\[ \forall \varepsilon > 0. \exists \delta > 0. \text{ input-change} \leq \delta \Rightarrow \text{ output-change} \leq \varepsilon \]
In general programs are not continuous. But they can be more continuous:

```
read sensor value x at time t;
compute “continuous” function y = f(x);
write output value y at time t+\delta:
```

Or less continuous:

```
read sensor value x;
if x ≤ c then y = f1(x)
  else y = f2(x);
```

. . . . . . . . . . . . . .
In general programs are not continuous. But they can be more continuous:

read sensor value \( x \) at time \( t \);
compute continuous function \( y = f(x) \);
write output value \( y \) at time \( t+d \);

Or less continuous:  
Better:

read sensor value \( x \);
if \( x \leq c \) then \( y = f_1(x) \)
else \( y = f_2(x) \); if \( x \leq c - \varepsilon \) then \( y = f_1(x) \);
if \( x \geq c + \varepsilon \) then \( y = f_2(x) \)
else \( y =\

\frac{(f_2(c+\varepsilon)-f_1(c-\varepsilon))(x-c+\varepsilon)/2\varepsilon + f_1(c-\varepsilon)}{\varepsilon} + f_1(c-\varepsilon)\

\[\text{[Majumdar et al., Gulwani et al.]}\]
3.2 Robust Systems

A Robustness as Mathematical Continuity:

- small input changes cause small output changes
- only possible in a quantitative framework

∀ ε>0. ∃ δ>0. input-change ≤ δ ⇒ output-change ≤ ε

Example of a Robustness Theorem [AHM]:
If discountedBisimilarity(A,B) > 1 - ε,
then ∀w : |A(w) – B(w)| < f(ε).
3.2 Robust Systems

A Robustness as Mathematical Continuity:
- small input changes cause small output changes
- only possible in a quantitative framework

B Robustness w.r.t. Faulty Assumptions:
- few environment mistakes cause few system mistakes
- ratio of system to environment mistakes as quantitative quality measure

[Greimel et al. TODAY]
3.3 Resource Interfaces

- Component interfaces expose resource requirements (e.g. time, memory, power).

- Interfaces are compatible if their combined requirements do not exceed the available resources.

- If the requirements are dynamic, then compatibility can be solved as a graph game with quantitative objectives.

[Chakrabarti et al.]
Max Value Constraint

minimizer

maximizer

node limit = 20
Max Value Constraint

node limit = 20

minimizer
maximizer
Sum Constraint

Path limit = -15

Minimizer:
- A
- B
- C
- D
- E
- F
- G
- H

Maximizer:
- 5
- 9
- 15
- 19
- 99
- 59
Sum Constraint

Path limit = 10

minimizer
maximizer
3.4 System Reliability

- assuming x% of input values are valid, y% of output values should be valid (limavg)

- hardware faulty, but can be replicated

- compiler ensures specified reliability through replication

[Ghosal et al.]
3.4 System Reliability

a: ok
b: fail

Limit-average value:

- aaaaaaaaaa... 1
- aaabaaabaaab... 3/4
- abaabaaab... 0

Want reliability of $1 - 10^{-x}$.
Conclusions

- “Quantitative” is more than “timed” and “probabilistic.”
- We need to move from boolean correctness criteria to quantitative system preference metrics.
- We have interesting point solutions, but no convincing overall framework.