Model Checking for tcc Calculus Documentation

Release 1.0

Jaime E. Arias Almeida

November 14, 2012

CONTENTS

Contents:

CHAPTER

ONE

FORMULA

This module contains the class to describe a temporal formula.

class formula.**Formula**(*data*)

This class represents a temporal formula.

Parameters **data** (*Dictionary.*) – Structure representing the temporal formula.

Example

$\phi = \Diamond(\mathtt{in} = \mathtt{true} \wedge \neg \circ (\mathtt{x} = 2))$

```
>>> from formula import *
>>> phi = Formula({"<>": {"^":{"":"in=true","~":{"o":"x=2"}}}})
```

Note: Logic operators are represented by the following symbols:

- Globally : []
- Future : <>
- Next : o
- Negation : ~
- Or : v
- And : ^

---

getConnective()

Return the main connective of the formula.

Returns A string representing the main connective of the formula.

Return type String.

Example

$\phi = \Diamond(\mathtt{in} = \mathtt{true} \wedge \neg \circ (\mathtt{x} = 2))$ $getConnective(\phi) = \Diamond$

```
>>> from formula import *
>>> phi = Formula({"<>": {"^":{"":"in=true","~":{"o":"x=2"}}}})
>>> phi.getConnective()
'<>'
```

getConsistentPropositions()

Returns the consistent propositions of a formula.

Returns A structure representing the consistent proposition of the formula.

Return type Dictionary.

Example

$\phi = (x = 2)$ $consistentPropositions(\phi) = \neg(x = 1)$

```
>>> from formula import *
>>> phi = Formula({"": "x=2"})
>>> phi.getConsistentPropositions()
{'~': 'x=1'}
```

getFormula()
Returns the formula.

Returns A structure representing the formula.

Return type Dictionary.

Example

```
>>> from formula import *
>>> phi = Formula({"<>": {"^":{"":"in=true","~":{"o":"x=2"}}}})
>>> phi.getFormula()
{'<>': {'^': {'': 'in=true', '~': {'o': 'x=2'}}}}
```

getNegation()
Returns the negation of the formula.

Returns The negation of the formula.

Return type Formula.

Example

$\phi = \circ(\mathtt{x} = 2)$ $\neg\phi = \neg \circ (\mathtt{x} = 2)$

```
>>> from formula import *
>>> phi = Formula({"o":"x=2"})
>>> negPhi = phi.getNegation()
>>> negPhi.getFormula()
{'~': {'o': 'x=2'}}
```

getPropositionRules()
Returns the consistent propositions of all propositions in the implementation.

Example

Returns A dictionary containing as key a proposition, and value all possible propositions that are consistent.

Return type Dictionary.

```
>>> from formula import *
>>> phi = Formula({"<>": {"^":{"":"in=true","~":{"o":"x=2"}}}})
>>> phi.getPropositionRules()
{'x=1': {'~': 'x=2'}, 'x=2': {'~': 'x=1'}}
```

getSubFormulas()
Returns the subformulas attached to a binary operator.

Returns A list containing the subformulas.

Return type List.

Example

$\alpha = (\texttt{in} = \texttt{true}) \wedge \neg \circ (\texttt{x} = 2)$ $\phi = (\texttt{in} = \texttt{true})$ $\psi = \neg \circ (\texttt{x} = 2)$

```
>>> from formula import *
>>> alpha = Formula({"^":{"":"in=true","~":{"o":"x=2"}}})
>>> subformulas = alpha.getSubFormulas()
>>> for subformula in subformulas:
...     print subformula.getFormula()
{'': 'in=true'}
{'~': {'o': 'x=2'}}
```

getValues()

Returns the formula without the outermost unary operator.

Returns A structure representing the formula without the outermost unary operator.

Return type Dictionary

Example

$\phi = \Diamond(\texttt{in} = \texttt{true} \wedge \neg \circ (\texttt{x} = 2))$ $getValues(\phi) = (\texttt{in} = \texttt{true}) \wedge \neg \circ (\texttt{x} = 2)$

```
>>> from formula import *
>>> phi = Formula({"<>": {"^":{"":"in=true","~":{"o":"x=2"}}}})
>>> phi.getValues()
{'^': {'': 'in=true', '~': {'o': 'x=2'}}}
```

isBasic()

Checks if the formula is a basic formula (i.e. proposition or it has $\circ$ as main connective)

Returns `True` if the formula is a basic formula or `False` otherwise.

Return type Boolean.

Example

```
>>> from formula import *
>>> phi = Formula({"o":"x=2"})
>>> phi.isBasic()
True
```

isNegativeFormula()

Returns if the formula has $\neg$ as main connective.

Returns `True` if the formula has $\neg$ as main connective or `False` otherwise.

Return type Boolean.

Example

```
>>> from formula import *
>>> phi = Formula({"~":{"o":"x=2"}})
>>> phi.isNegativeFormula()
True
```

isNegativeNext()

Checks if the formula is of the form $\neg \circ \phi$.

Returns `True` if the formula is of the form $\neg \circ \phi$ or `False` otherwise.

Return type Boolean.

Example

```
>>> from formula import *
>>> phi = Formula({"~": {"○":"x=2"}})
>>> phi.isNegativeNext()
True
```

isProposition()

Checks if the formula is a proposition.

Returns `True` if the formula is a proposition or `False` otherwise.

Return type Boolean.

Example

```
>>> from formula import *
>>> phi = Formula({"":"x=2"})
>>> phi.isProposition()
True
```

CHAPTER

TWO

CLOSURE

This module contains the functions neccesary to generate the closure of a temporal formula

closure.**getClosure**(*formula*, *closure*)

Function that generates the closure of a temporal formula.

Parameters

- **formula** (*Formula*) – Temporal formula that we want to find the closure
- **closure** (*List*) – Empty list to store the subformulas of the closure

Example

```
>>> from closure import *
>>> phi = Formula({"<>": {"^":{"":"in=true","~":{"o":"x=2"}}}})
>>> closure = []
>>> getClosure(phi,closure)
>>> for formula in closure:
...     print formula.getFormula()
...
{'<>': {'^': {'': 'in=true', '~': {'o': 'x=2'}}}}
{'~': {'<>': {'^': {'': 'in=true', '~': {'o': 'x=2'}}}}}
{'o': {'<>': {'^': {'': 'in=true', '~': {'o': 'x=2'}}}}}
{'~': {'o': {'<>': {'^': {'': 'in=true', '~': {'o': 'x=2'}}}}}}
{'o': {'~': {'<>': {'^': {'': 'in=true', '~': {'o': 'x=2'}}}}}}
{'^': {'': 'in=true', '~': {'o': 'x=2'}}}
{'~': {'^': {'': 'in=true', '~': {'o': 'x=2'}}}}
{'': 'in=true'}
{'~': 'in=true'}
{'o': 'x=2'}
{'~': {'o': 'x=2'}}
{'o': {'~': 'x=2'}}
{'': 'x=2'}
{'~': 'x=2'}
```

Note: This function is based on the conditions shown in the section 6.1 of the thesis document.

CHAPTER

THREE

MODEL CHECKING GRAPH

This module contains the necessary functions to generate a model checking graph.

modelCheckingGraph.**getAllAtoms**(*closure*)

Returns all possible atoms of the closure.

Parameters **closure** (List of Formula) – Closure of a formula.

Returns List of all atoms of the closure.

Return type List of lists of Formula.

Example

```
>>> from closure import *
>>> from modelCheckingGraph import *
>>> phi = Formula({"<>": {"^":{"":"in=true","~":{"o":"x=2"}}}})
>>> closure = []
>>> getClosure(phi,closure)
>>> atoms = getAllAtoms(closure)
>>> for index, atom in enumerate(atoms):
...     print "Atom " + str(index) + ":"
...     for formula in atom:
...             print formula.getFormula()
...
Atom 0:
{'o': {'<>': {'^': {'': 'in=true', '~': {'o': 'x=2'}}}}}
{'': 'in=true'}
{'o': 'x=2'}
{'': 'x=2'}
{'<>': {'^': {'': 'in=true', '~': {'o': 'x=2'}}}}
{'~': {'^': {'': 'in=true', '~': {'o': 'x=2'}}}}
Atom 1:
{'~': {'o': {'<>': {'^': {'': 'in=true', '~': {'o': 'x=2'}}}}}}
{'': 'in=true'}
{'o': 'x=2'}
{'': 'x=2'}
{'o': {'~': {'<>': {'^': {'': 'in=true', '~': {'o': 'x=2'}}}}}}
{'~': {'<>': {'^': {'': 'in=true', '~': {'o': 'x=2'}}}}}
{'~': {'^': {'': 'in=true', '~': {'o': 'x=2'}}}}
...
```

See Also:

closure.getClosure()

Note: This function is based on the algorithm shown in [MP95].

modelCheckingGraph.**getBasicFormulas**(*closure*)

Returns the basic formulas (i.e. propositions or formulas with ○ as main connective) of the closure.

Parameters **closure** (List of Formula) – Closure of a formula.

Returns List of basic formulas of the closure.

Return type List of Formula.

Example

```
>>> from closure import *
>>> from modelCheckingGraph import *
>>> phi = Formula({"<>": {"^":{"":"in=true","~":{"o":"x=2"}}}})
>>> closure = []
>>> getClosure(phi,closure)
>>> basicFormulas = getBasicFormulas(closure)
>>> for formula in basicFormulas:
...     print formula.getFormula()
...
{'o': {'<>': {'^': {'': 'in=true', '~': {'o': 'x=2'}}}}}
{'': 'in=true'}
{'o': 'x=2'}
{'': 'x=2'}
```

See Also:

closure.getClosure()

modelCheckingGraph.**getModelCheckingAtoms**(*tcc_structure*, *atoms*)

modelCheckingGraph.**getModelCheckingGraph**(*tcc_structure*, *model_checking_atoms*)

modelCheckingGraph.**getNoBasicFormulas**(*closure*)

Returns the formulas of the closure that are not basic formulas.

Parameters **closure** (List of Formula) – Closure of a formula.

Returns List of formulas of the closure that are not basic formulas.

Return type List of Formula.

Example

```
>>> from closure import *
>>> from modelCheckingGraph import *
>>> phi = Formula({"<>": {"^":{"":"in=true","~":{"o":"x=2"}}}})
>>> closure = []
>>> getClosure(phi,closure)
>>> noBasicFormulas = getNoBasicFormulas(closure)
>>> for formula in noBasicFormulas:
...     print formula.getFormula()
...
{'<>': {'^': {'': 'in=true', '~': {'o': 'x=2'}}}}
{'^': {'': 'in=true', '~': {'o': 'x=2'}}}
```

See Also:

closure.getClosure()

modelCheckingGraph.**isConsistent**(*formula*, *atom*)

Checks if a formula is consistent with the set of formulas in an atom.

Parameters

- **formula** (Formula) – Formula
- **atom** (List of Formula.) – List of consistent formulas representing an atom of the closure.

Returns True if the formula is consistent with the set of formulas in the atom or False otherwise.

Return type Boolean

Example

```
>>> from closure import *
>>> from modelCheckingGraph import *
>>> phi = Formula({"<>": {"^":{"":"in=true","~":{"o":"x=2"}}}})
>>> closure = []
>>> getClosure(phi,closure)
>>> atoms = getAllAtoms(closure)
>>> isConsistent(Formula({'': 'x=1'}), atoms[0])
False
```

Note: This function is based on the conditions shown in the defintion 6.1 of the thesis document.

---

modelCheckingGraph.**isInAtom**(*formula*, *atom*)

Checks if a formula is in an atom.

Parameters

- **formula** (*Dictionary*) – Structure representing a formula.
- **atom** (List of Formula.) – List of consistent formulas representing an atom of the closure.

Returns True if the formula is in atom or False otherwise.

Return type Boolean

Example

```
>>> from closure import *
>>> from modelCheckingGraph import *
>>> phi = Formula({"<>": {"^":{"":"in=true","~":{"o":"x=2"}}}})
>>> closure = []
>>> getClosure(phi,closure)
>>> atoms = getAllAtoms(closure)
>>> isInAtom({'': 'in=true'}, atoms[0])
True
```

modelCheckingGraph.**isNextState**(*nextFormulas*, *nextAtom*)

modelCheckingGraph.**propositionConsistent**(*formula*, *atom*)

modelCheckingGraph.**searchFormulas**(*formulas*, *connective*)

Returns the formulas that have a particular main connective.

Parameters

- **formulas** (List of Formula) – List of formulas.
- **connective** (*String*) – The main connective.

Returns List containing the formulas that have the main connective.

Return type List

Example

```
>>> from modelCheckingGraph import *
>>> list = [Formula({'o': 'x=2'}), Formula({'~': {'o': 'x=2'}}), Formula({'o': {'~': 'x=2'}})]
>>> result = searchFormulas(list,'o')
>>> for formula in result:
...     print formula.getFormula()
...
{'o': 'x=2'}
{'o': {'~': 'x=2'}}
```

CHAPTER

FOUR

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

BIBLIOGRAPHY

[MP95] Zohar Manna and Amir Pnueli. Temporal Verification of Reactive Systems: Safety. Springer-Verlag New York, Inc., 1995.

PYTHON MODULE INDEX

INDEX