GR(1) Synthesis for LTL Specification Patterns

README with materials overview

Shahar Maoz and Jan Oliver Ringert

Research Paper Abstract

Reactive synthesis is an automated procedure to obtain a correct-by-construction reactive system from its temporal logic specification. Two of the main challenges in bringing reactive synthesis to software engineering practice are its very high worst-case complexity -- for linear temporal logics (LTL) it is double exponential in the length of the formula, and the difficulty of writing declarative specifications using basic LTL operators.

To address the first challenge, Piterman et al. have suggested the General Reactivity of Rank 1 (GR(1)) fragment of LTL, which has an efficient polynomial time symbolic synthesis algorithm. To address the second challenge, Dwyer et al. have identified 55 LTL specification patterns, which are common in industrial specifications and make writing specifications easier.

In this work we show that almost all of the 55 LTL specification patterns identified by Dwyer et al. can be expressed in the GR(1) fragment of LTL. Specifically, we present an automated, sound and complete translation of the patterns to the GR(1) form, which effectively results in an efficient reactive synthesis procedure for any specification that is written using the patterns.

We have validated the correctness of the catalog of GR(1) templates we have created. The work is implemented in our reactive synthesis environment. It provides positive, promising evidence, for the potential feasibility of using reactive synthesis in practice.

Overview of Files and Folders

- Folders with produced artifacts
  - **./ltl/**: contains files with LTL formulas of patterns
  - **./dbw/**: contains minimal DBW represented in LTL2DSTAR format
  - **./templates/**: contains generated templates of LTL specification patterns in GR(1) format
  - **./verification/**: contains generated verification conditions for templates to check using NuSMV
- **patternCatalog.pdf**: overview over all templates for supported patterns and their negations
- **DBW2GR1.jar**: tool to translate a deterministic Büchi automaton in LTL2DSTAR format to a GR(1) template
- Information on 3rd party tools of complete toolchain at end of this document

ltl Folder: Specification Patterns LTL Semantics

- The folder **ltl** contains files with the LTL semantics for all (negated) LTL specification patterns by Dwyer et al. Template files are numbered as they appear in patternCatalog.pdf and on the website by Dwyer et al. http://patterns.projects.cis.ksu.edu/documentation/patterns/ltl.shtml
  - files with the extension ltl3dra use the LTL syntax of LTL3DRA
  - files with the extension ltl2dstar use the LTL syntax of LTL2DSTAR

dbw Folder: Minimal Deterministic Büchi Word Automata

- The folder **dbw** contains minimal Deterministic Büchi Word automata (DBW) for all (negated) LTL specification patterns by Dwyer et al. that can be expressed as DBW.
  - 52 LTL specification patterns have a corresponding DBW
  - 41 negated LTL specification patterns have a corresponding DBW
  - DBW were created using outputs of LTL3DRA and LTL2DSTAR (see file extension, not all DBW for LTL2DSTAR were computed due to the sizes of intermediate artifacts)

templates Folder: Specification Pattern GR(1) Templates

- The folder **templates** contains GR(1) templates for all (negated) LTL specification patterns by Dwyer et al. that can be expressed as Deterministic Büchi Word (DBW) automata. Template files are numbered as they appear in patternCatalog.pdf and on the website by Dwyer et al. http://patterns.projects.cis.ksu.edu/documentation/patterns/ltl.shtml
  - 52 LTL specification patterns have a corresponding DBW and thus a GR(1) template.
  - 41 negated LTL specification patterns have a corresponding DBW and thus a GR(1) template.

Example Template

```
VAR -- auxiliary variables: states of DBW
  s : {S1, S2, bot};
INIT -- initial assignments: initial state
  s=S1;
TRANS -- safety this and next state
  ((s=S1  & ( r |  p | !q) & X s=S1) |
   (s=S1  & (!r & !p &  q) & X s=S2) |
   (s=S2  & (!r & !p     ) & X s=S2) |
   (s=S2  & ( r          ) & X s=bot)|
   (s=S2  & (!r &  p     ) & X s=S1) |
   (s=bot & (TRUE        ) & X s=bot));
LTLSPEC -- justice part: accepting states
  G F (s=S1 | s=S2);
```

auxiliary variables introduced to support pattern

GR(1) safety constraint referring to current and next state including parameters of pattern (here: p, q, and r)

GR(1) justice constraint

verification Folder: Templates Extended for Verification

- The folder **verification** contains the following files:
  - **`pattern##(neg)?.(ltl3dra|ltl2dstar)`** GR(1) templates extended with verification conditions to be checked by NuSMV
  - **`pattern##(neg)?.(ltl3dra|ltl2dstar).report`** verification results produced by NuSMV

- Files are created using the LTL to deterministic Rabin automaton translation of **`ltl3dra`** or **`ltl2dstar`** (indicated in file name).

Template Extended for Verification

```
VAR -- auxiliary variables: states of DBW
  s : {S1, S2, bot};
INIT -- initial assignments: initial state
  s=S1;
TRANS -- safety this and next state
  ((s=S1  & ( r |  p | !q) & X s=S1) |
   (s=S1  & (!r & !p &  q) & X s=S2) |
   (s=S2  & (!r & !p     ) & X s=S2) |
   (s=S2  & ( r          ) & X s=bot)|
   (s=S2  & (!r &  p     ) & X s=S1) |
   (s=bot & (TRUE        ) & X s=bot));
LTLSPEC -- justice part: accepting states
 (G F (state=S0 | state=S1)) <->
       (G(!(q&!r)|((p&!r)V(!r|(p&!r)))));
-- parameter values free (after initial)
CTLSPEC AG EX (!q & !r & !p);
CTLSPEC AG EX (q & !r & !p);
...
CTLSPEC AG EX (!q & r & p);
CTLSPEC AG EX (q & r & p);
 -- parameter values free (initial)
LTLSPEC !(!q & !r & !p);
LTLSPEC !(q & !r & !p);
...
LTLSPEC !(!q & r & p);
LTLSPEC !(q & r & p);
```

original template

equality of satisfaction of GR(1) justice and original LTL of pattern

$2^{|parameters|}$ checks that every combination parameter values is possible in every next state

$2^{|parameters|}$ checks that each combination of parameter values is impossible in the initial state

Verification Output of NuSMV

```
*** This is NuSMV 2.5.4 (compiled on Fri Oct 28 14:13:29 UTC 2011)
…
*** Copyright (c) 2010, Fondazione Bruno Kessler

-- specification AG (EX ((!q & !r) & !p))  is true
-- specification AG (EX ((q & !r) & !p))  is true
-- specification AG (EX ((!q & r) & !p))  is true
-- specification AG (EX ((q & r) & !p))  is true
-- specification AG (EX ((!q & !r) & p))  is true
-- specification AG (EX ((q & !r) & p))  is true
-- specification AG (EX ((!q & r) & p))  is true
-- specification AG (EX ((q & r) & p))  is true
-- specification (G F (state=S0 | state=S1)) <-> (G(!(q&!r)|((p&!r)V(!r|(p&!r)))))  is true
-- specification !((!q & !r) & !p)  is false
-- specification !((q & !r) & !p)  is false
-- specification !((!q & r) & !p)  is false
-- specification !((q & r) & !p)  is false
-- specification !((!q & !r) & p)  is false
-- specification !((q & !r) & p)  is false
-- specification !((!q & r) & p)  is false
-- specification !((q & r) & p)  is false
```

values of parameters should be unrestricted ($2^{|parameters|}$ CTL checks must succeed)

satisfaction of template and original LTL should coincide

values of parameters in initial state should be unrestricted (last $2^{|parameters|}$ checks must fail)

DBW2GR1.jar: Translation of Büchi automata to GR(1)

- **Input**: deterministic Büchi automaton as created by DBAMinimize in LTLDSTAR format
- **Output**: GR(1) template and GR(1) templates extended for verification
- **Usage**:

```
java -jar DBW2GR1.jar [InputFile]
                [-v LTLfile]? > [OutputFile]
```

  - parameter -v creates extended template for verification (when using this option DBW2GR1 requires a file with the LTL semantics of the pattern in NuSMV syntax)
  - the LTL syntax used by LTL3DRA is the same as the one used by NuSMV, the syntax used by LTL2DSTAR is different

DBW2GR1 Example Usage

- Generate GR(1) template:

```
 java -jar DBW2GR1.jar ./dbw/pattern09.ltl3dra >
    ./templates/pattern09.smv
```

- Generate GR(1) template for verification:

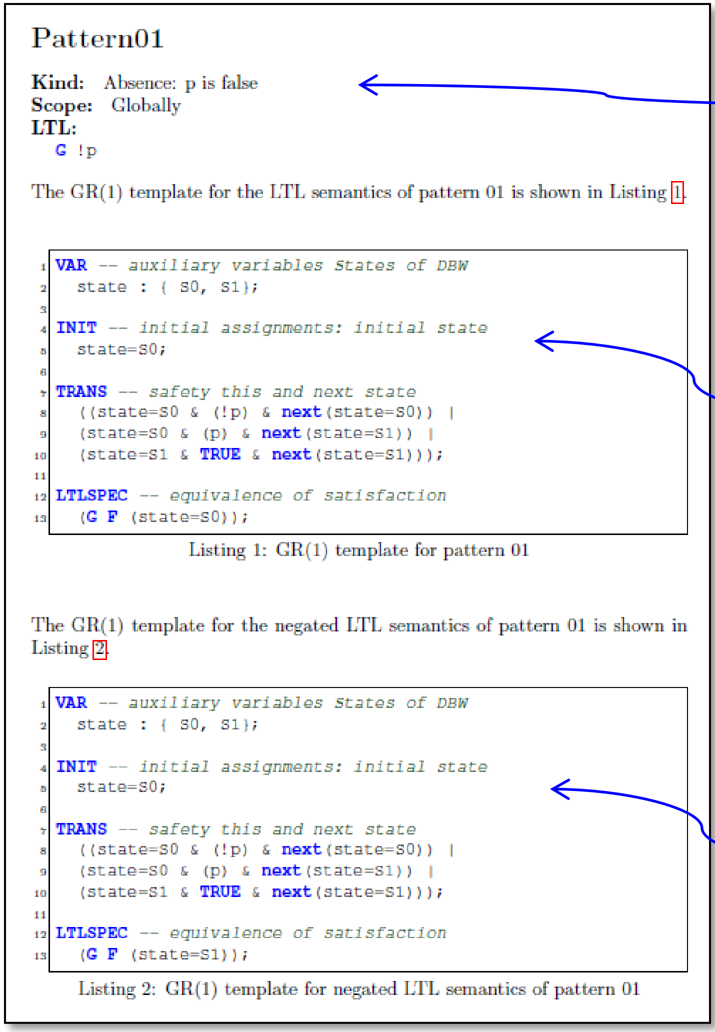
```
java -jar DBW2GR1.jar ./dbw/pattern09.ltl3dra
        -v ./ltl/pattern09.ltl3dra > ./verification/pattern09.ltl3dra
```

  - requires the files

    `./dbw/pattern09.ltl3dra` (DBW in DSTAR format)

    `./ltl/pattern09.ltl3dra` (LTL semantics of pattern in NuSMV syntax)

patternCatalog.pdf: GR(1) Template Catalog



Pattern01

Kind: Absence: p is false
Scope: Globally
LTL:

```
G !p
```

The GR(1) template for the LTL semantics of pattern 01 is shown in Listing 1.

```
VAR -- auxiliary variables States of DBW
  state : { S0, S1};

INIT -- initial assignments: initial state
  state=S0;

TRANS -- safety this and next state
  ((state=S0 & (!p) & next(state=S0)) |
  (state=S0 & (p) & next(state=S1)) |
  (state=S1 & TRUE & next(state=S1)));

LTLSPEC -- equivalence of satisfaction
  (G F (state=S0));
```

Listing 1: GR(1) template for pattern 01

The GR(1) template for the negated LTL semantics of pattern 01 is shown in Listing 2.

```
VAR -- auxiliary variables States of DBW
  state : { S0, S1};

INIT -- initial assignments: initial state
  state=S0;

TRANS -- safety this and next state
  ((state=S0 & (!p) & next(state=S0)) |
  (state=S0 & (p) & next(state=S1)) |
  (state=S1 & TRUE & next(state=S1)));

LTLSPEC -- equivalence of satisfaction
  (G F (state=S1));
```

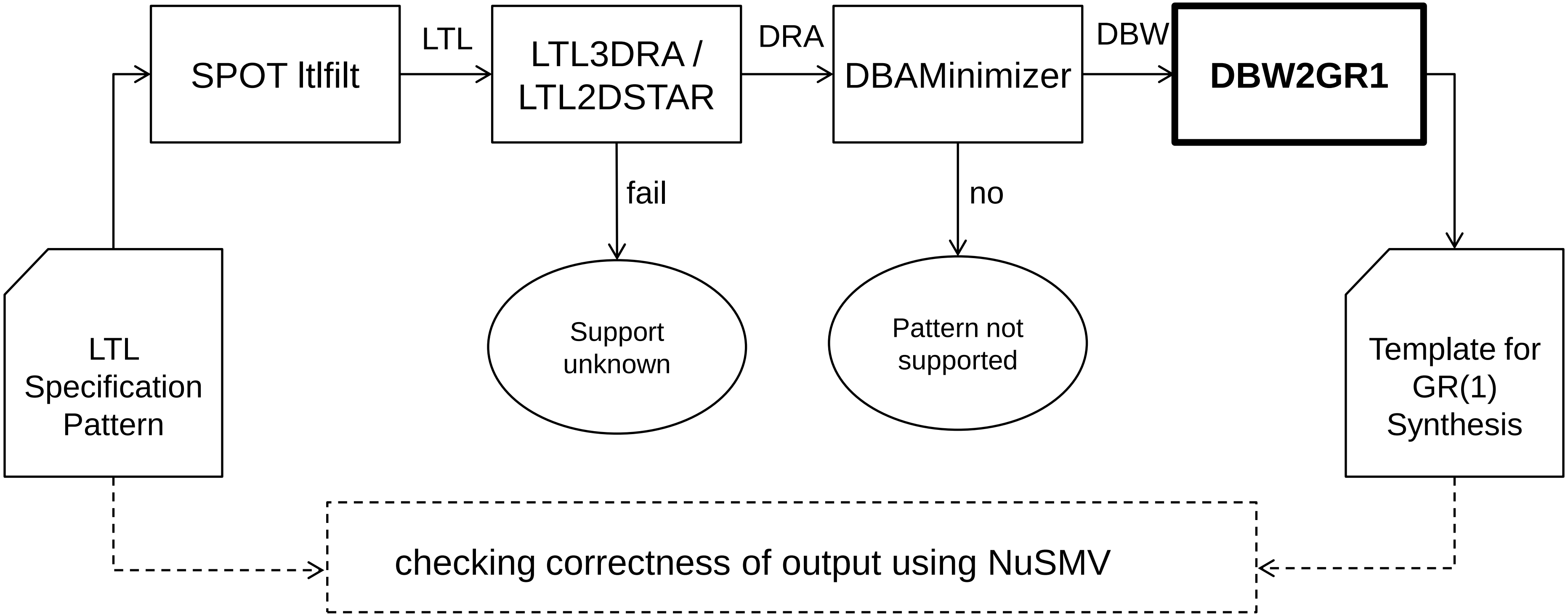
Listing 2: GR(1) template for negated LTL semantics of pattern 01

number, kind, scope, and LTL of specification pattern

template to express LTL semantics in GR(1) format, if a deterministic Büchi automaton for the LTL semantics exists

template to express **negated** LTL semantics in GR(1) format, if a deterministic Büchi automaton for the **negated** LTL semantics exists

Toolchain Overview



LTL Specification Pattern

SPOT ltlfilt

LTL

LTL3DRA / LTL2DSTAR

DRA

DBAMinimizer

DBW

DBW2GR1

fail

no

Support unknown

Pattern not supported

Template for GR(1) Synthesis

checking correctness of output using NuSMV

3rd Party Tools in Toolchain

- DBAMimimizer
  - https://www.react.uni-saarland.de/tools/dbaminimizer/
  - DBA Minimizer package, version from 25/11/2011
- LTL2DSTAR
  - http://www.ltl2dstar.de/
  - ltl2dstar version 0.5.2
- LTL3DRA
  - http://sourceforge.net/projects/ltl3dra/
  - ltl3dra version v0.2.1
- NuSMV
  - http://nusmv.fbk.eu/
  - NuSMV version 2.5.4
- SPOT ltlfilt
  - http://spot.lip6.fr/userdoc/ltlfilt.html
  - ltlfilt included in SPOT version 1.2.6

Scripts to Execute Toolchain in Cygwin Bash

```
# create deterministic rabin automata from LTL of patterns using LTL3DRA and LTL2DSTAR
for i in ltl/*.ltl3dra; do ./ltl3dra_v0.2.1.exe -L -F "$i" > dra/"${i/ltl}"; done
for i in ltl/*.ltl2dstar; do ./ltl2dstar_v0.5.2.exe --ltl2nba=spin:./ltl3ba.exe "$i" dra/"${i/ltl}"; done

# create minimal deterministic Büchi automata from Rabin automata using DBAMinimizer
for i in dra/*.ltl2dstar; do ./minimize.py "$i" &> dbw/"${i/dra}"; done

# create SMV code for verification from Büchi automata
for i in dbw/*.ltl3dra; do java -jar DBW2GR1.jar ./"$i" -v ./ltl/"${i/dbw}" > chk/"${i/dbw}"; done
for i in dbw/*.ltl2dstar; do java -jar DBW2GR1.jar ./"$i" -v ./ltl/"${{i/dbw}/ltl2dstar/ltl3dra}" > chk/"${i/dbw}"; done

# check generated SMV modules and put results in report files for inspection
for i in chk/*.ltl3dra; do NuSMV -dcx "$i">"$i".report; done
for i in chk/*.ltl2dstar; do NuSMV -dcx "$i">"$i".report; done

# create GR(1) templates from Büchi automata
for i in dbw/*.ltl3dra; do java -jar DBW2GR1.jar ./"$i"> templates/"${i/dbw}".smv; done
for i in dbw/*.ltl2dstar; do java -jar DBW2GR1.jar ./"$i"> templates/"${i/dbw}".smv; done
```