vTSL – A Formally Verifiable DSL for Specifying Robot Tasks

Christian Heinzemann and Ralph Lange

Robert Bosch GmbH
Corporate Sector Research and Advance Engineering
Renningen, Germany
[firstname.lastname]@de.bosch.com
vTSL – A Formally Verifiable DSL for Specifying Robot Tasks

Motivation

Planning

Executive

Skills

SLAM  LineDetect  PathControl

Camera
vTSL – A Formally Verifiable DSL for Specifying Robot Tasks

Motivation

Skills
- SLAM
- LineDetect
- PathControl
- Camera

Planning

Executive

... camera.config(ACTIVE)
lineDetect.config(ACTIVE)
pathControl.driveLinear(0.3)
...
if (FOUND_LINE) then
...
end
vTSL – A Formally Verifiable DSL for Specifying Robot Tasks

Motivation

Planning

Executive

Skills

SLAM  LineDetect  PathControl  Camera  …
vTSL – A Formally Verifiable DSL for Specifying Robot Tasks

Motivation

Planning

Executive

Skills

How to verify?

Platform-specific constraints and rules

Custom (sub)tasks

© Robert Bosch GmbH 2018. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.
vTSL – A Formally Verifiable DSL for Specifying Robot Tasks

Language Concepts and Example

while(retries < maxRetries)
  [Status moveStatus, Status lineStatus, Status distStatus] = call/or*
    MoveLinear(speed), DetectLine(<< ... >>), DistanceMonitor(100);
  if (lineStatus == success) then
    setGroundCameraState(false);
    return << ... >>;
  else if (distStatus == success) then
    // failed to detect line -> drive back and try again
    speed = -speed;
  end
  retries++;
end

setGroundCameraState(false);

Task-tree semantics · C-like expressiveness · synchronous programming interfacing with skill-layer (ROS) · source code generation · verifiability
vTSL – A Formally Verifiable DSL for Specifying Robot Tasks

Translation and Model Checking

Implemented using
- JetBrains MPS
- Spin (Promela)

- Task tree modeling
  - code generation
  - access

- Component stubs
  - refers to

- Formal model for verification
  - verify against

- C++ code for task-tree library

- Constraint specification DSL
  - translate to

- Formal specification
  - translate to
vTSL – A Formally Verifiable DSL for Specifying Robot Tasks

Christian Heinzemann and Ralph Lange

Robert Bosch GmbH
Corporate Sector Research and Advance Engineering
Renningen, Germany
[firstname.lastname]@de.bosch.com
vTSL – A Formally Verifiable DSL for Specifying Robot Tasks

Motivation

FindLoadingBay
MoveLinear
DetectLine
DistanceMonitor

How to verify?

Platform-specific constraints and rules

Requirements and Key Ideas

Verifiability
Fully automated transformation for verification tool – omit feature if not translatable

Expressiveness
Modern programming language features desired – resemblance with C/C++

Interfacing with skill layer
Interface with ROS and provide abstract replacement model for skill component behaviors

Task Tree Modeling

Task Tree Modeling

Translation and Model Checking

Implemented using
• JetBrains MPS
• Spin (Promela)

Implemented using
• JetBrains MPS
• Spin (Promela)

C++ code for task-tree library

Constraint specific, DSL

Formal model for verification

Formal specification

Task tree modeling
code generation

Component stubs

access

refers to

translate to

translate to

verify against
vTSL – A Formally Verifiable DSL for Specifying Robot Tasks

Related Work

  - Most (59 of 137) DSLs have control and event handling on architectural level on focus
  - http://corlab.github.io/dslzoo/

  - The only verification approach for task trees – but only for activation and synchronization

  - Script language for programming manipulation tasks
  - Supports semiautomatic verification with Coq

- Various verification approaches for (hierarchical) FSMs from CPS community and robotics
VTS – A Formally Verifiable DSL for Specifying Robot Tasks

**Verifiability**

Fully automated transformation for verification tool – omit feature if not translatable

**Task tree semantics**

Principle of decomposition as in Task Description Language or Hierarchical Task Networks

**Expressiveness**

Modern programming language features desired – resemblance with C/C++

**Synchronous Programming**

Concurrency inspired by Céu, but not the strong determinism of synchronous languages

**Interfacing with skill layer**

Interface with ROS and provide abstract replacement model for skill component behaviors

**Source code generation**

Task trees should be directly generatable into implement-ta-tion for the robot

vTSL – A Formally Verifiable DSL for Specifying Robot Tasks

**Action FindLoadingBay**

```
action FindLoadingBay::<...>>
    behavior normal
    setGroundCameraState(true);
    uint8 const maxRetries = 3;
    uint8 retries = 0;
    double speed = 0.2;
    while(retries < maxRetries)
        if (lineStatus == success) then
            setGroundCameraState(false);
            return::<...>>;
        end if
        if (distStatus == success) then
            speed = -speed;
        end if
        retries++;
    end while
    setGroundCameraState(false);
```

**Action DetectLine**

```
action DetectLine::<...>>
    behavior normal
    setGroundCameraState(false);
    fail:
end
```

**Action MoveLinear**

```
action MoveLinear(double speed)
    behavior forward execute if speed >= 0
    float64 speedMax;
    connect RobotBase.execute(as speedMaxExec with queue size 1);
    while(true)
        speedMax.data = speed;
    end
    write speedStatus to RobotBase.linearSpeed;
    // wait for robot base controller
    read speedMaxExec;
    end
end on abort
    float64 speedSlow;
    speedSlow.data = 0.5;
    write speedStatus to RobotBase.linearSpeed;
end
```

Formula to be verified by the model checker
vTSL – A Formally Verifiable DSL for Specifying Robot Tasks

ROS Message

@ROS Message, Package Resource Name: iros_robot
import Image

struct LineDetectMsg {
    Image rawImage;
    uint8 blueComponent;
}

ROS Service

@ROS Service, Package Resource Name: std_srvs/SetBool

service SetBool {
    struct Request {
        boolean data; // e.g. for hardware enabling / disabling
    }

    struct Response {
        boolean success; // indicate successful run of triggered service
        string message; // informational, e.g. for error messages
    }
}

Component Stub

skill component GroundCamera {

    Advertised Topics:
        topic lineDetectMsg lineDetectTopic:

    Offered Services:
        service SetBool activateCamera:

    initialization
        boolean cameraState = false;
        SetBool.Request activateCamQuery:
        SetBool.Response activateCamResp:
        end

    execute behavior
        choose
            [activateCamQuery, activateCamResp] = receive GroundCamera.activateCamera();
            cameraState = activateCamQuery.data;
            activateCamResp.success = true;
            reply GroundCamera.activateCamera(activateCamResp, true);
            or
            (cameraState == true):
            LineDetectMsg lineMsg:
            choose
                lineMsg.blueComponent = 59;
                or
                lineMsg.blueComponent = 61;
            end
            write lineMsg to GroundCamera.lineDetectTopic;
            end
}
vTSL – A Formally Verifiable DSL for Specifying Robot Tasks

Translation and Model Checking

Implemented using

- JetBrains MPS
- Spin (Promela)

Promela Model

```c
byte runningAction;
byte readyQueue[DISPATCHER_QUEUE_SIZE];
byte blockedQueue[DISPATCHER_QUEUE_SIZE];
byte abortSubaction[2];

proctype FindLoadingBay()
{
  byte childPid, childPid2;
  // wait for run signal
  (runningAction == _pid);

  // user defined action code starts here!
  _findLoadingBay_setGroundCameraState(true);
  #define maxRetries 3
  byte retries = 0;

do
  :: retries < maxRetries -> {
    mtype distStatus;
    childPid = run gen__lambda__IROS_Gen_FindLoadingBay_1();
    childPid2 = run IROS_Gen_DistanceMonitor(200);
    callParallelSubactions(childPid, childPid2);
    awaitParallelOBSubactionReturn(childPid, childPid2, distStatus);
    (runningAction == _pid);
    if
        :: IROS_Gen_FindLoadingBay_normal_lineStatus == SUCCESS -> {
            _IROS_Gen_FindLoadingBay_setGroundCameraState(false);
            ActionReturn(SUCCESS);
        }
        :: else -> { ... }
    fi;
    retries++;
  } else -> { break; }
  od;

  _findLoadingBay_setGroundCameraState(false);
  ActionReturn(FAILED);

  // user defined action code ends here!
  unless
    :: (abortSubaction[0] == _pid || abortSubaction[1] == _pid);
    (runningAction == _pid);
    (ActionReturn(ABORTED);
  )
  ;
  ActionReturn(SUCCESS);
}
```

Language Dependencies
PML file  The Spin Model Checker (spinroot.com)

```
$ spin -search -l MyTaskTree.pml
pan:1: assertion violated (BenchmarkComponent_simpleTopic.registeredClients<1) (at depth 286)
pan: wrote MyTaskTree.pml.trail
pan: reducing search depth to 287

(Spin Version 6.4.8 -- 2 March 2018)
+ Partial Order Reduction

Full statespace search for:
  never claim - (none specified)
  assertion violations +
  cycle checks - (disabled by -DSAFETY)
  invalid end states +

State-vector 668 byte, depth reached 286, errors: 1
  97 states, stored
  7 states, matched
  104 transitions (= stored+matched)
  212 atomic steps
hash conflicts: 0 (resolved)
unreached in proctype vtsl__dispatcher
  MyTaskTree.pml:618, state 8, "abortRootAction = 0"
  MyTaskTree.pml:617, state 9, "abortSubaction[0] = rootActionPid"
  MyTaskTree.pml:624, state 16, "-end-
  (3 of 16 states)
unreached in proctype vtsl__externalEventHandler
  MyTaskTree.pml:240, state 8, "vtsl_i = (vtsl_i+1)"
  MyTaskTree.pml:247, state 16, "vtsl_i = 0"

...  

MyTaskTree.pml:1112, state 106
  MyTaskTree.pml:1120, state 110, "-end-
  (24 of 110 states)

pan: elapsed time 0.02 seconds
pan: rate 4850 states/second
```
vTSL – A Formally Verifiable DSL for Specifying Robot Tasks

**Experiment 1: Scalability w.r.t. Tree Size**

![Graph showing scalability w.r.t. Tree Size]

**Experiment 2: Scalability w.r.t. Component Count**

![Graph showing scalability w.r.t. Component Count]
vTSL – A Formally Verifiable DSL for Specifying Robot Tasks

We are seeking out for an academic partner to join forces!