Verification Case Studies with ObjectCheck

Fei Xie

(Joint work with James C. Browne, Robert P. Kurshan, and Vladimir Levin)
Presentation Outline

• Overview and Architecture of ObjectCheck
• Modeling and Verification of a TinyOS Run-time Image with ObjectCheck
• More Case Studies
• Summary and Future Work
• Work Built on ObjectCheck
ObjectCheck Overview

• An integrated development, validation, and model-checking environment for software system designs;
  – System designs are specified in xUML;
  – xUML is an executable subset of UML;
• Developed in conjunction with
  – FormalCheck (Robert P. Kurshan, et. al.);
  – SDLCheck (Vladimir Levin and Husnu Yenigun);
  – **Goal**: Software/hardware co-design and co-verification;
    • **Sub-goal**: Model checking of software system designs in xUML.
Executable UML (xUML)

- Has well-defined *Execution Semantics*;
- Utilizes *UML Action Semantics* recently adopted by OMG;
- Can be compiled to procedural codes;
- Tools provided by:
  - Project Technologies;
  - Kennedy Carter;
  - Hyperformix (SES);
  - …
Architecture and Workflow of ObjectCheck

Property Specification Interface ➔ Property ➔ xUML-to-S/R Translator ➔ S/R Query ➔ COSPAN Model Checker

xUML IDE ➔ xUML Model ➔ Error Report Generator

Error Visualizer ➔ Error Report ➔ Error Track

Designer ➔ Property ➔ xUML Model ➔ Error Report Generator
Presentation Outline

• Overview and Architecture of ObjectCheck
  • Modeling and Verification of a TinyOS Run-time Image with ObjectCheck
• More Case Studies
• Summary and Future Work
• Work Built on ObjectCheck
Case Study on TinyOS

• A run-time image of TinyOS, a component-based operating system for networked sensors;
• xUML models for TinyOS components have been constructed from their C source codes to enable:
  – Model-driven development on design level;
  – Software/hardware co-design and co-verification;
• Two properties are to be checked:
  – *Repeated transmission* on physical network;
  – *No consecutive 0’s* in any sequence-number sequence.
More on TinyOS

• An OS for networked sensors that have
  – Limited computation ability and energy supply;
  – High concurrency requirements;
  – Diversities in designs and usages;
  – High reliability requirement;

• Is component-based
  – Run-time images are specialized for different sensors;
  – Only necessary components are selected and composed.

• More information -- [http://webs.cs.berkeley.edu/tos](http://webs.cs.berkeley.edu/tos)
Step-by-Step Demonstration

Designer

Property Specification Interface

Property

xUML IDE

xUML Model

Error Visualizer

Error Report

xUML-to-S/R Translator

Error Report Generator

S/R Query

S/R Model

Error Track

COSPAN Model Checker
Buf = Data;

Generate TQS2: Post_Task (1),

4. Photo_Intr

S03: P_Intr
(SO_ID, Data)

S06: Post_Task_Ret
(SO_ID)

5. Intr_Ret

S08: Back_to_Idle
(SO_ID)

Generate P4: P_Intr_Ret (1);

Generate S08: Back_to_Idle (SO_ID);

1. Idle

S01: C_Intr
(SO_ID)

S08: Back_to_Idle
(SO_ID)

2. Starting_Photo

Generate P1: P_Start (1),

S02: P_Start_Ret
(SO_ID)

3. Starting_Ret

Generate CL2: C_Intr_Ret (1);

Generate S08: Back_to_Idle (SO_ID),
if (Full == FALSE) {
    if (Tail == 5) Tail = 0;
    else Tail = Tail + 1;

    if (Emp == FALSE) {
        Generate SOT1: Dispatch (1);
        if (Head == 5) Head = 0;
        else Head = Head + 1;
    }

    if (Full == TRUE) Full = FALSE;
    if (Head == Tail) Emp = TRUE;
}

if (Full == FALSE) {
    if (Tail == 5) Tail = 0;
    else Tail = Tail + 1;

    if (Emp == TRUE) Emp = FALSE;
    if (Tail == Head) Full = TRUE;
}

Generate SOT6: Post_Task_Ret (1);
Generate TQS4: Back_to_Idle (1);

5. Post_Rcvd (TQ_ID)  TQS2: Post_Task (TQ_ID)  3. Dispatching  TQS1: Schedule (TQ_ID)  1. Idle  TQS2: Post_Task (TQ_ID)  2. Post_Rcvd

TQS3: Dispatch_Ret (TQ_ID)
TQS3: Dispatch_Ret (TQ_ID)

4. Dispatch_Ret
Generate SOT4: T_Ret (1);
Generate TQS4: Back_to_Idle (1);
Buf = Data;

Generate T082: Post_Task (1);

Generate P4: P_Intr_Ret (1);

Generate S03: Back_to_Idle (S0_ID);

Generate S08: Back_to_Idle (S0_ID);

Generate S06: Post_Task_Ret (S0_ID);

Generate S02: P_Start_Ret (S0_ID);

Generate CL2: C_Intr_Ret (1);

Generate S00: Back_to_Idle (S0_ID);
Sensor_Output_Lifecycle.grf
enzo.cs.utexas.edu% objectbench
Setting up Objectbench Query Language ...
Setup completed

Please input the name of the tob file

(type "quit" to abort the generation process)

=>

..working/snb.tob
project s2n
consistof
domain s2n

    syntype Interrupt_Type = integer
        constants [0 4] default 0
    endsyntype

    arraytype Integer_Array = Integer
        6
    endarraytype

0 0
subsystem Sensor_to_Network
    0 0
        internal
            object Hardware
                HW
                Id 'HW_ID'
                Interrupt_Type 'Choice'.
        item
            HW_Running Yes 1
                ('HW_ID', 1)
                ('Choice', 0)
        enditem

1 0

    event HW1: C_Ret ( id key )
    event HW2: A_Ret ( id key )
    event HW3: R_Ret ( id key )
    event HW4: T_Ret ( id key )
    event HW5: Loop ( id key )

    state HW_Running HW(SDL)_Running
        0 0
        Choice = ANY("Interrupt_Type")
        if (Choice == 0) Generate CL1: HW_C_Intr (1)
        if (Choice == 1) {
            if (ADC1.On == TRUE) Generate ADC3: HW_A_Intr (1)
Step-by-Step Demonstration
DECLARE RFM_Pending <<SYSTEM S2N/BLOCK SENSOR_TO_NETWORK/PROCESS RFM>> PENDING;
DECLARE HW_Choice <<SYSTEM S2N/BLOCK SENSOR_TO_NETWORK/PROCESS HARDWARE>> CHOICE;

/* Properties */
Repeatedly(#RFM_Pending == 1);
Repeatedly(#RFM_Pending != 1);

/* Assumptions */
AssumeRepeatedly(#HW_Choice == 0);
AssumeRepeatedly(#HW_Choice == 1);
AssumeRepeatedly(#HW_Choice == 2);
AssumeRepeatedly(#HW_Choice == 3);
AssumeRepeatedly(#HW_Choice == 4);
Step-by-Step Demonstration

- Property Specification Interface
- Property
- xUML-to-S/R Translator
- S/R Query
- xUML IDE
- xUML Model
- S/R Model
- Error Visualizer
- Error Report
- Error Report Generator
- Error Track
- COSPAN Model Checker
enzo.cs.utexas.edu% tcb2sr -ob -at -query query1.qry sn.tob
tob2sr is executing...
tob2sr: Total time spent = 250 msec.
enzo.cs.utexas.edu%
# define __DummySignal 0

/* incoming signals of processes and channels */

# define Process_HARDWARE_HW5 1
# define Process_HARDWARE_HW4 2
# define Process_HARDWARE_HW3 3
# define Process_HARDWARE_HW2 4
# define Process_HARDWARE_HW1 5
type Process_HARDWARE_Signals : ( __DummySignal, Process_HARDWARE_HW5, Process_HARDWARE_HW4, Process_HARDWARE_HW3, Process_HARDWARE_HW2, Process_HARDWARE_HW1 )

# define Process_CLOCK_CL2 1
# define Process_CLOCK_CL1 2
type Process_CLOCK_Signals : ( __DummySignal, Process_CLOCK_CL2, Process_CLOCK_CL1 )

# define Process_ADC_ADC3 1
# define Process_ADC_ADC2 2
# define Process_ADC_ADC1 3
type Process_ADC_Signals : ( __DummySignal, Process_ADC_ADC3, Process_ADC_ADC2, Process_ADC_ADC1 )

# define Process_PHOTO_P4 1
# define Process_PHOTO_P3 2
# define Process_PHOTO_P2 3
# define Process_PHOTO_P1 4

# define Process_SENSOR_OUTPUT_SO6 1
# define Process_SENSOR_OUTPUT_SO3 2
Process_TASK_QUEUE_N_QTQ1 ? (__Sender = p_Process_HARDWARE) |
__DummySignal
)

selvar __InputSig : Process_TASK_QUEUE_N_Signals
asgn __InputSig := __SigArray[__SlotToBeConsumed]
end BufferProcess_TASK_QUEUE_N

#endif

#if ! defined(__MODEL_LOCAL)

/*-------------------------------*/
/* QUERY PART */
/*-------------------------------*/

#define DIRECTSUMof 2

#include <QRY.h>

/*-------------------------------*/
/* USER QUERY */
/*-------------------------------*/

/* Query variable definitions */

#define __QV__RFM_Pending (Process_RF.M.V_PENDING)
#define __QV__HW_Choice (Process_HARDWARE.V_CHOICE)

/* Properties */
monitor __Property__1 : Repeatedly_ (0, (__QV__RFM_Pending=1))
monitor __Property__2 : Repeatedly_ (1, (__QV__RFM_Pending"=1))

/* Assumptions from the query file */
monitor __Assumption__1 : AssumeRepeatedly_ ((__QV__HW_Choice=0))
monitor __Assumption__2 : AssumeRepeatedly_ ((__QV__HW_Choice=1))
monitor __Assumption__3 : AssumeRepeatedly_ ((__QV__HW_Choice=2))
monitor __Assumption__4 : AssumeRepeatedly_ ((__QV__HW_Choice=3))
monitor __Assumption__5 : AssumeRepeatedly_ ((__QV__HW_Choice=4))
#endif
Step-by-Step Demonstration

- Property Specification Interface
- xUML IDE
- Error Visualizer
- Property
- xUML Model
- Error Report
- xUML-to-S/R Translator
- S/R Model
- Error Report Generator
- S/R Query
- Error Track
- COSPAN Model Checker
Mon Apr  7 17:16:31 CDT 2003
enzo.cs.utexas.edu [SunOS 5.8 sun4u]: /v/filer1a/v0q031/feixie/project/OBChek/k/demo/tos/working1
cospan -bq1 -Kt -#status -#dupstvars -#time -D__MODEL_LOCAL sn__query1.sr -D__QUERY_LOCAL sn__query1.sr
cospan: Version 8.23.206 (Bell Laboratories) 1 Feb 2002
       Single pass for option -q1
cospan: Version 8.23.206 (Bell Laboratories) 1 Feb 2002
+ time sr_F -I/projects/formalcheck/COSPAN/SUN/include -#status -#dupstvars -I/projects/formalcheck/COSPAN/SUN/include -b -#reduction -Kt -#status -#dupstvars -D__MODEL_LOCAL sn__query1.sr -D__QUERY_LOCAL sn__query1.sr -#qtree -#status -#dupstvars -#reduction -#status -#dupstvars -#qtree
sr_F: -#bddversion=dl
Status: Begin parsing at 0 sec 0 megabytes.

sn__query1.sr: Mon Apr  7 17:16:17 2003
/projects/formalcheck/COSPAN/SUN/include/QRY.h: Mon Mar 18 10:42:35 2002
/projects/formalcheck/COSPAN/SUN/include/QRY+.h: Mon Mar 18 10:42:35 2002
Status: Begin checks and tree rewrites at 0.1 sec 0.73728 megabytes.

sn__query1.rf: list entry count:
   93 pruned, 211 active, 0 freed by reduction
39 data variables declared or with width >= -#databits=4
144 selection/local variables
0 resized variables
67 bounded state variables: 2.58e53 states
0 unbounded state variables
5 boolean cysets
2 boolean recurs
2 free selection/local variables: 50 selections/state
0 pausing processes
Status: Begin tree to bdd-expr translation at 0.96 sec 0.90112 megabytes.
   (-b: 142 macro'd selvars, 2 retained selvars)

Status: Begin bdd-expr to local bdd translation at 1 sec 3.53075 megabytes 1164 bdd nodes.

sn__query1.sr: Synchronous model

Status: Begin global bdds at 15.8 sec 8.43776 megabytes 27089 bdd nodes.
   2 initial states.
Status: Begin forward search at 21.02 sec 8.43776 megabytes 39310 bdd nodes.
   4.64949e+06 states reached.
State set generations 2258
Status: Begin usage statistics at 989.83 sec 68.78 megabytes 1975701 bdd nodes.

Bounded stvar range coverage: 67 variables, 64.43% average coverage
   24 enumerated and boolean: 20 values of 2 variables unreach
   43 integer: 30 variables with unreach values; 46.08% average cove

   worst coverage: 0.10% for 1 variables
   32 bounded variables (47.76%) have unreach values

Status: Begin cycle check method 1 at 990.37 sec 68.78 megabytes 1975701 bdd
   nodes.
Status: Begin forward envelope computation at 1018.41 sec 68.78 megabytes 197
   5828 bdd nodes.
1975828 bdd nodes, 1017.66 seconds, 65.2493 megabytes

sn__query1.sr: Task performed!

real 17:00.0
user 16:58.6
sys 1.0
see query1.log.out
enzo.cs.utexas.edu%
DECLARE HW_Amount <<SYSTEM S2N/BLOCK SENSOR_TO_NETWORK/PROCESS HARDWARE>> AMOUNT;
DECLARE RFM_RxF <<SYSTEM S2N/BLOCK SENSOR_TO_NETWORK/PROCESS RFM>> RXF;
DECLARE RFM_Tx <<SYSTEM S2N/BLOCK SENSOR_TO_NETWORK/PROCESS RFM>> TX;
DECLARE RFM_Amount <<SYSTEM S2N/BLOCK SENSOR_TO_NETWORK/PROCESS RFM>> AMOUNT;

/* Logic Propositions */
DECLARE HW_Amount <<SYSTEM S2N/BLOCK SENSOR_TO_NETWORK/PROCESS HARDWARE>> AMOUNT;
DECLARE RFM_RxF <<SYSTEM S2N/BLOCK SENSOR_TO_NETWORK/PROCESS RFM>> RXF;
DECLARE RFM_Tx <<SYSTEM S2N/BLOCK SENSOR_TO_NETWORK/PROCESS RFM>> TX;
DECLARE RFM_Amount <<SYSTEM S2N/BLOCK SENSOR_TO_NETWORK/PROCESS RFM>> AMOUNT;

/* Properties */
Never ((#RFM_Data == 0) AND (#RFM_Buf == 0) AND #RFM_Transmitting);

/* Assumptions */
AssumeRepeatedly(#HW_Amount == 0);
AssumeRepeatedly(#HW_Amount == 1);
AssumeRepeatedly(#HW_Amount == 2);
AssumeRepeatedly(#HW_Amount == 3);
AssumeRepeatedly(#HW_Amount == 4);
enzo.cs.utexas.edu% tcb2sr -ob -at -query query2.qry sn.tob

tob2sr is executing...
tob2sr: Total time spent = 233 msec.
Mon Apr  7 17:16:55 CDT 2003
enzo.cs.utexas.edu [SunOS 5.8 sun4u]: /v/filer1a/v0q031/feixie/project/OBCheck/k/demo/tos/working2
cosp an -bq1 -Kt -#status -#dupstvars -#time -D__MODEL_LOCAL sn__query2.sr -D__QUERY_LOCAL sn__query2.sr

cosp an: Version 8.23.206 (Bell Laboratories)  1 Feb 2002
  Single pass for option -q1
  cosp an: Version 8.23.206 (Bell Laboratories)  1 Feb 2002
+ time sr_E -I/projects/formalcheck/COSPAN/SUN/include -#status -#dupstvars -I/projects/formalcheck/COSPAN/SUN/include -b -#reduction -Kt -#status -#dupstvars -D__MODEL_LOCAL sn__query2.sr -D__QUERY_LOCAL sn__query2.sr -#qtree -#status -#dupstvars -#reduction -#status -#dupstvars -#qtree
sr_E: -#bddversion=d1
Status: Begin parsing at 0 sec 0 megabytes.
sn__query2.sr: Mon Apr  7 17:16:53 2003
/projects/formalcheck/COSPAN/SUN/include/QRY.h: Mon Mar 18 10:42:35 2002
/projects/formalcheck/COSPAN/SUN/include/QRY++.h: Mon Mar 18 10:42:35 2002
Status: Begin checks and tree rewrites at 0.11 sec 0.73728 megabytes.
sn__query2.rf: list entry count:
  60 pruned, 243 active, 0 freed by reduction
  39 data variables declared or with width >= -#databits=4
  153 selection/local variables
  1 resized variables
  90 bounded state variables: 1.82e126 states
  0 unbounded state variables
  6 boolean cysets
  0 boolean recurs
  2 free selection/local variables: 50 selections/state
  1 variable reference clippings, 0 expression clippings
Status: Begin tree to bdd-expr translation at 1.04 sec 0.94208 megabytes.
   (-b: 151 macro’d selvars, 2 retained selvars)

Status: Begin bdd-expr to local bdd translation at 1.16 sec 3.56352 megabytes
   1652 bdd nodes.

sn_query2.sr: Synchronous model
Status: Begin global bdds at 16.07 sec 8.81459 megabytes 30959 bdd nodes.
   1 initial states.
Status: Begin forward search at 23.74 sec 8.81459 megabytes 68301 bdd nodes.
Stop at error:
   No transition enabled in process .__Property__1
3916 states reached.
State set generations 248
Status: Begin error track at 59.72 sec 11.092 megabytes 164852 bdd nodes.
Status: Begin usage statistics at 64.58 sec 11.1002 megabytes 164852 bdd nodes.
Bounded stvar range coverage: 89 variables, 47.11\% average coverage
   24 enumerated and boolean: 30 values of 4 variables unreachable
   65 integer: 52 variables with unreached values; 30.50\% average coverage
   worst coverage: 0.05\% for 10 variables
56 bounded variables (62.92\%) have unreached values
156202 bdd nodes, 64.91 seconds, 7.81517 megabytes
sn_query2.sr: Task failed (tree).

real 1:06.6
user 1:06.1
sys 0.1
see query2.log.out
enzo.cs.utexas.edu
Step-by-Step Demonstration

Property Specification Interface

- Property

xUML IDE

- xUML Model

Error Visualizer

- Error Report

Error Report Generator

- Error Track

xUML-to-S/R Translator

- S/R Query

- S/R Model

COSPAN Model Checker
Output line too long.
"": 37, 20   Error in error track file!!!
tr2tob: There are 248 states in the error track (1 in the post mortem part)
T2otr: See the file "sn__query2.otr" for the error track...
PROCESS HARDWARE is initially at STATE START
<<SYSTEM S2N / BLOCK SENSOR TO NETWORK / PROCESS HARDWARE>>CHOICE=0
PROCESS CLOCK is initially at STATE CL(SDL_IDLE)
PROCESS ADC is initially at STATE ADC(SDL_IDLE)
<<SYSTEM S2N / BLOCK SENSOR TO NETWORK / PROCESS ADC>>ON=0
<<SYSTEM S2N / BLOCK SENSOR TO NETWORK / PROCESS ADC>>READING=0
PROCESS PHOTO is initially at STATE P(SDL_IDLE)
<<SYSTEM S2N / BLOCK SENSOR TO NETWORK / PROCESS PHOTO>>DATA=0
PROCESS SENSOR_OUTPUT is initially at STATE SO(SDL_IDLE)
<<SYSTEM S2N / BLOCK SENSOR TO NETWORK / PROCESS SENSOR_OUTPUT>>BUF=0
<<SYSTEM S2N / BLOCK SENSOR TO NETWORK / PROCESS SENSOR_OUTPUT>>DATA=0
PROCESS RFM is initially at STATE RFM(SDL_IDLE)
<<SYSTEM S2N / BLOCK SENSOR TO NETWORK / PROCESS RFM>>PENDING=0
<<SYSTEM S2N / BLOCK SENSOR TO NETWORK / PROCESS RFM>>BUF=1
<<SYSTEM S2N / BLOCK SENSOR TO NETWORK / PROCESS RFM>>DATA=0
PROCESS GENERIC_COMM is initially at STATE GC(SDL_IDLE)
<<SYSTEM S2N / BLOCK SENSOR TO NETWORK / PROCESS GENERIC_COMM>>BUF=0
<<SYSTEM S2N / BLOCK SENSOR TO_NETWORK / PROCESS GENERIC_COMM>>DATA=0
PROCESS INT_TO_RFМ is initially at STATE IR(SDL_IDLE)
<<SYSTEM S2N / BLOCK SENSOR TO NETWORK / PROCESS INT_TO_RFМ>>DATA=0
PROCESS TASK_QUEUE_S is initially at STATE TQS(SDL_IDLE)
<<SYSTEM S2N / BLOCK SENSOR TO NETWORK / PROCESS TASK_QUEUE_S>>FULL=0
<<SYSTEM S2N / BLOCK SENSOR TO NETWORK / PROCESS TASK_QUEUE_S>>EMP=1
<<SYSTEM S2N / BLOCK SENSOR TO NETWORK / PROCESS TASK_QUEUE_S>>HEAD=0
<<SYSTEM S2N / BLOCK SENSOR TO NETWORK / PROCESS TASK_QUEUE_S>>TAIL=0
PROCESS SQ_TASK is initially at STATE SOT(SDL_IDLE)

--More--{(4%)}
Step-by-Step Demonstration

Property Specification Interface

Designer

Property

xUML IDE

xUML Model

Error Visualizer

Error Report

xUML-to-S/R Translator

S/R Query

Error Report Generator

S/R Model

Error Track

COSPAN Model Checker
enzo.cs.utexas.edu%  ls
query2.log    sn_query2.M    sn_query2.mnp    sn_query2.tr
query2.log.out sn_query2.T    sn_query2.otr    zrun2
query2.qry    sn_query2.U    sn_query2.rf
sn.tob        sn_query2.err    sn_query2.sr
enzo.cs.utexas.edu% otr2sim sn_query2.otr sn_query2.sim HARDWARE
enzo.cs.utexas.edu%
/*************** Interleaving orders: *****************/

delay 0;
HARDWARE(1).Seq_S = 1;
HARDWARE(1).CHOICE = 0;

delay 1;
HARDWARE(1).Seq_T = 15;

delay 1;
CLOCK(1).Seq_S = 0;

delay 1;
CLOCK(1).Seq_T = 0;

delay 1;
SENSOR_OUTPUT(1).Seq_S = 0;

delay 1;
SENSOR_OUTPUT(1).Seq_T = 0;

delay 1;
PHOTO(1).Seq_S = 0;

delay 1;
PHOTO(1).Seq_T = 0;

delay 1;
ADC(1).Seq_S = 0;

--More--(16%)
Generate HW6: Start (HW_ID);
delay Seq_T;

delay Seq_S;

/* CHOICE = ANY("Interrupt_Type"); */

if (CHOICE == 0) Generate OL1: HW_O_Intr (1);

if (CHOICE == 1) {
    if (ADC(1).On == TRUE) Generate ADC3: HW_A_Intr (1);
    else {
        Generate HW5: Loop (HW_ID);
        CHOICE = 1;
    }
}

if (CHOICE == 2) {
    if (RFM(1).P ending == TRUE) Generate RFM3: HW_R_Intr (1);
    else {
        Generate HW5: Loop (HW_ID);
        CHOICE = 2;
    }
}

if (CHOICE == 3) {
    if (TASK_QUEUE_S(1).Exp == FALSE) Generate TQS1: Schedule (1);
    else {
        Generate HW5: Loop (HW_ID);
        CHOICE = 3;
    }
}

if (CHOICE == 4) {
    if (TASK_QUEUE_N(1).Exp == FALSE) Generate TQH1: Schedule (1);
    else {
        Generate HW5: Loop (HW_ID);
        CHOICE = 4;
    }
}
Generate HW6: Start (HW_ID);
delay Seq_T;

if (CHOICE == 2) {
    if (FPM(1).Pending == TRUE) Generate FPM3: HW_R_Intr (1);
    else {
        Generate HW5: Loop (HW_ID);
        CHOICE = 2;
    }
}
Statistics from Model Checking “Repeated Output” Property

• Four model checking runs with different combinations of reduction algorithms
  – Start at the same time on the same host;
  – The host is a SUN with 8 CPUs and 2GB memory.

<table>
<thead>
<tr>
<th>SPOR</th>
<th>SMC</th>
<th>Memory Usage</th>
<th>Time Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>Off</td>
<td>596M</td>
<td>19370S</td>
</tr>
<tr>
<td>Off</td>
<td>On</td>
<td>80M</td>
<td>1384S</td>
</tr>
<tr>
<td>On</td>
<td>Off</td>
<td>596M</td>
<td>17438S</td>
</tr>
<tr>
<td>On</td>
<td>On</td>
<td>102M</td>
<td>1379S</td>
</tr>
</tbody>
</table>

Conclusion: SMC helps and SPOR doesn’t.
Presentation Outline

- Overview and Architecture of ObjectCheck
- Modeling and Verification of a TinyOS Run-time Image with ObjectCheck
- More Case Studies
- Summary and Future Work
- Work Built on ObjectCheck
Model Checking of NASA robot Controller

• A typical control-intensive embedded system;
• Conducted by Natasha Shyrigina using ObjectCheck;
  – 37 properties were checked.
  – 22 properties were successfully checked.
  – 6 bugs were found.
### Properties to be Model checked

<table>
<thead>
<tr>
<th>N</th>
<th>Property</th>
<th>Robotic Description</th>
<th>Formal Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EventuallyAlways(p \implies \neg p) ((p=1))</td>
<td>Eventually the robot control terminates</td>
<td>Eventually permanently (p=1)</td>
</tr>
<tr>
<td>2</td>
<td>Always(\neg U \implies \neg p) ((\neg \forall \theta \implies \neg p) \land \theta \geq 1)</td>
<td>The program terminates when its either completes the task or reaches the state where there is no solution for the fault recovery</td>
<td>(p=0) holds at any execution of the program until occurrence of either (\theta = 1) or the combination of (\theta = 1) and (\theta = 1)</td>
</tr>
<tr>
<td>3</td>
<td>Always(\neg U \implies \neg p) ((\neg \forall \theta \implies \neg p) \land \theta \geq 1)</td>
<td>The program terminates when its either completes the task or reaches the state where there is no solution for the fault recovery</td>
<td>(p=0) holds at any execution of the program until occurrence of either (\theta = 1) or the combination of (\theta = 1) and (\theta = 1)</td>
</tr>
<tr>
<td>4</td>
<td>Always (r = 1 \implies \theta = 1)</td>
<td>Whenever the EE is in the &quot;Follow-up desirable trajectory&quot; state than the EE is in the &quot;Valid&quot; state</td>
<td>At any time during execution of the program when (r = 1) than (\theta = 1)</td>
</tr>
<tr>
<td>5</td>
<td>Always(\neg U \implies \neg p) ((\neg \forall \theta \implies \neg p) \land \theta \geq 1)</td>
<td>Fault recovery is executed when any of the joint angles does not satisfy the allowed limits</td>
<td>(s=0) holds at any execution until any of the following does not hold (\theta = \text{max(\theta)}) OR (\theta &lt; \text{min(\theta)}) OR (\theta = \text{max(\theta)}) OR (\theta &lt; \text{min(\theta)})</td>
</tr>
<tr>
<td>6</td>
<td>Always (s=1 \land \theta = 1 \land \theta = 1)</td>
<td>Fault recovery is always executed for joints that resides in their most recent base position</td>
<td>At any execution of the program it is always the case that (s=1) and all of the following hold (\theta = \text{max(\theta)}) \land (\theta = \text{min(\theta)})</td>
</tr>
<tr>
<td>7</td>
<td>Never(\neg U \implies \neg p) ((\neg \forall \theta \implies \neg p) \land \theta \geq 1)</td>
<td>When fault recovery is called, the EE can not move to a new position until fault is resolved</td>
<td>It is never the case that (r = 1) holds when (s=1) holds. This condition can be terminated by (p=0)</td>
</tr>
<tr>
<td>8</td>
<td>Eventually(\forall \theta \implies \neg p)</td>
<td>If an obstacle is reached by the EE then the obstacle avoidance procedures is performed</td>
<td>It is always the case that if (\text{EE} \geq \text{Obstacle}) OR (\text{EE} &lt; \text{Obstacle}) otherwise it is in the future (s=1) holds</td>
</tr>
<tr>
<td>9</td>
<td>Deadlock Freedom</td>
<td>The program does not have deadlocks</td>
<td>The model does not have deadlocks</td>
</tr>
</tbody>
</table>

### Table 4.5: Verification properties (continued)

<table>
<thead>
<tr>
<th>N</th>
<th>Property</th>
<th>Robotic Description</th>
<th>Formal Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Never ((c_p &gt; \text{L_min} \lor c_p &lt; \text{L_max}) \land r=1)</td>
<td>The robot never operates outside of allowed workspace</td>
<td>It is never the case that (c_p &gt; \text{L_min} \lor c_p &lt; \text{L_max}) holds and (r=1) holds</td>
</tr>
<tr>
<td>11</td>
<td>Never ((c_p &gt; 0) \land \theta &lt; \text{L_min} \lor c_p &gt; \text{L_max})</td>
<td>The program always performs computations for an actual robot the general description of a robot is not used</td>
<td>At any execution of the program the current value of (c_p) never exceeds the value of the (\theta) parameter</td>
</tr>
<tr>
<td>12</td>
<td>Never ((c_p &gt; 0) \land \theta &lt; \text{L_min} \lor c_p &gt; \text{L_max})</td>
<td>No command to move the EE is scheduled before an initial EE position is computed</td>
<td>It is never the case that (r=1) holds until (s=1) holds</td>
</tr>
<tr>
<td>13</td>
<td>Never ((c_p &gt; 0) \land \theta &lt; \text{L_min} \lor c_p &gt; \text{L_max})</td>
<td>The EE does not proceed to a new position until an acknowledgement from the JCH is received</td>
<td>At any execution (r=1) holds does not hold until either (c_p = 0) and (s=1) holds or (s=0) holds</td>
</tr>
<tr>
<td>14</td>
<td>Eventually(\forall \theta \implies c_p = 0)</td>
<td>The robot arm always follows the specified trajectory</td>
<td>At any execution of the program (c_p = 0) holds</td>
</tr>
<tr>
<td>15</td>
<td>Eventually(\forall \theta \implies c_p = 0)</td>
<td>Chained fault recovery is not permitted (if the fault recovery did not complete for an instance of the robot configuration, the fault recovery for a different robot configuration instance is not allowed)</td>
<td>At any execution of the future (p=0) holds</td>
</tr>
<tr>
<td>16</td>
<td>(r=1 \lor \neg \text{Obstacle})</td>
<td>(r=1) and (c_p = \text{Obstacle}) AND (r=1) AND (c_p = \text{Obstacle}) are never possible at the same time</td>
<td>At any execution (r=1) and (c_p = \text{Obstacle}) AND (r=1) AND (c_p = \text{Obstacle}) are never possible at the same time</td>
</tr>
<tr>
<td>17</td>
<td>(c_p = 0 \land \theta = 1)</td>
<td>Only validated solutions of TrajConfigurations are used for the optimization of the robot control</td>
<td>At any time when (c_p = 0) holds</td>
</tr>
</tbody>
</table>
Model Checking of Online Ticket Sale System

• A typical commercial transaction system;
• Presented at FASE 2002;
• Focus: Integrated state space reduction.
An Online Ticket Sale System (Class Diagram)
An Online Ticket Sale System
(A State Model)

if (Agent_Status[0] == TRUE) && (Agent_Status[1] == TRUE) {
    choice = ANY("A_ID_TYPE");
    Agent_Status[choice - 1] = FALSE;
    Generate A1: Assignment (choice, Served_0, Served_0, Requested_Amount);
} else if ((Agent_Status[0] == TRUE) && (Agent_Status[1] == FALSE)) {
    Agent_Status[0] = FALSE;
    Generate A1: Assignment (1, Served_0, Served_0, Requested_Amount);
} else if ((Agent_Status[0] == FALSE) && (Agent_Status[1] == TRUE)) {
    Agent_Status[1] = FALSE;
    Generate A1: Assignment (2, Served_0, Served_0, Requested_Amount);
} else
    Generate D5: Try_Later (Served_0, Served_0);

Generate D3: Back_to_Idle (Dispatcher_ID);

if (Sold_Out == TRUE) {
    Generate C4: No_Ticket (0_ID, 0_ID);
    Generate D3: Back_to_Idle (Dispatcher_ID);
} else {
    Served_0 = 0_ID, Served_0 = 0_ID;
    Requested_Amount = Num;
    Generate D5: Loop (Dispatcher_ID);
}
Some Verification Statistics of Online Ticket Sale System

• Verification of a liveness property
  – After an agent is assigned to a customer, eventually the agent will be released.

• Statistics related to state space reductions

<table>
<thead>
<tr>
<th>SPOR</th>
<th>SMC</th>
<th>Memory Usage</th>
<th>Time Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>Off</td>
<td>Out of Memory</td>
<td>-</td>
</tr>
<tr>
<td>Off</td>
<td>On</td>
<td>113.73M</td>
<td>44736.5S</td>
</tr>
<tr>
<td>On</td>
<td>Off</td>
<td>17.3M</td>
<td>6668.3S</td>
</tr>
<tr>
<td>On</td>
<td>On</td>
<td>74.0M</td>
<td>1450.3S</td>
</tr>
</tbody>
</table>
Presentation Outline

- Overview and Architecture of ObjectCheck
- Modeling and Verification of a TinyOS Run-time Image with ObjectCheck
- More Case Studies
- Summary and Future Work
- Work Built on ObjectCheck
Summary and Future Work

• **ObjectCheck**
  – Integrates industrial software development environments and model checkers with research tools;
  – Provides comprehensive automation for development, validation, and model checking of xUML models;
  – Has enabled verification of non-trivial software system designs modeled in xUML.

• **Future work is focused on**
  – State space reduction capability of ObjectCheck;
  – Hardware/software co-design and co-verification.
Related Work

• Most closely related work
  – UML Model Checking toolset from University of Michigan;
  – vUML tool from Åbo Akademi University;

• There is also related work on model checking of statecharts with different semantics.
Additional Information

- [http://www.cs.utexas.edu/users/ObjectCheck](http://www.cs.utexas.edu/users/ObjectCheck)
- **Selected publications:**
  - Fei Xie and James C. Browne. Verified Systems by Composition from Verified Components. Submitted for review.
Presentation Outline

- Overview and Architecture of ObjectCheck
- Modeling and Verification of a TinyOS Run-time Image with ObjectCheck
- More Case Studies
- Summary and Future Work
- Work Built on ObjectCheck
Work Built on ObjectCheck

• An integrated state space reduction framework;
• Integration of model checking into component-based development of software.
Integrated State Space Reduction Framework

- Verification Task
  - User-Driven State Space Reduction

- Verification Subtasks
  - Reduced xUML Model
  - Reduced xUML Level Query

- Basic Model Checking Process
  - xUML-to-S/R Translation
    - S/R Model
    - S/R Level Query
    - Model Checking with COSPAN
      - Success Report / Error Track

- Techniques
  - Decomposition
  - Abstraction
  - Symmetry Reduction
  - Partial Order Reduction
  - Symbolic Verification
  - Localization Reduction
Reduction Steps for Checking $P_0$

Step 1: Symmetry Reduction
Customers, Dispatcher
Agents, Ticket Server

Step 2: Decomposition
Customers, Dispatcher
Agents, Ticket Server

Step 3: Symmetry Reduction
Customers
Dispatcher
Agents, Ticket Server

Step 4: Decomposition
Agents, Ticket Server

Step 5: Case Splitting
Agents
Ticket Server

Step 6: Symmetry Reduction
Agents
Ticket Server
Evaluation of User-driven State Space Reduction

• Directly model checking $P_0$ on OTSS
  – Two customer instances and two agent instances;
  – SPOR and SMC are both applied.
  – **Memory usage:** 152.79M
  – **Time usage:** 16273.7S

• Memory and time usages for discharging subtasks at the leaf nodes of the reduction tree.

<table>
<thead>
<tr>
<th></th>
<th>$P_{21}$</th>
<th>$P_{22}$</th>
<th>$P_{41}$</th>
<th>$P_{42}$</th>
<th>$P_{43}$</th>
<th>$P_{44}$</th>
<th>$P_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>0.30M</td>
<td>0.95M</td>
<td>0.28M</td>
<td>0.29M</td>
<td>0.28M</td>
<td>0.29M</td>
<td>0.35M</td>
</tr>
<tr>
<td>Time</td>
<td>0.02S</td>
<td>1.81S</td>
<td>0.01S</td>
<td>0.04S</td>
<td>0.01S</td>
<td>0.04S</td>
<td>0.63S</td>
</tr>
</tbody>
</table>
Integration of Model Checking into Component-based Development

• Temporal properties of a component are
  – Established with assumptions on the environment of the component;
  – Verified under these assumptions and then packaged with the component.

• Selecting a component for reuse considers not only its functionality but also its temporal properties.

• Properties of a composed component are verified by reusing verified properties of its sub-components and applying compositional reasoning.
Sensor Component
Properties of Sensor Component

Properties:
Repeatedly (Output);
After (Output) Never (Output) UntilAfter (OP_Ack);
After (Done) Eventually (Done_Ack);
Never (Done_Ack) UntilAfter (Done);
After (Done_Ack) Never (Done_Ack) UntilAfter(Done);

Assumptions:
After (Output) Eventually (OP_Ack);
Never (OP_Ack) UntilAfter (Output);
After (OP_Ack) Never (OP_Ack) UntilAfter (Output);
After (Done) Never (Done) UntilAfter (Done_Ack);
Repeatedly (C_Intr);
After (C_Intr) Never (C_Intr + A_Intr + S_Schd) UntilAfter (C_Ret);
After (ADC.Pending) Eventually (A_Intr);
After (A_Intr) Never (C_Intr + A_Intr + S_Schd) UntilAfter (A_Ret);
After (STQ.Empty = FALSE) Eventually (S_Schd);
After (S_Schd) Never (C_Intr + A_Intr + S_Schd) UntilAfter (S_Ret);