PAT Vision: ***Pervasive Model Checking***

- *Model Checking as Planning/Problem-Solving/Scheduling/Services*
- *Wide application domains, including Real-Time and Probabilistic systems.*

Model checking as planning/problem-solving

```
//@@Sliding Game@@
//The following models the sliding game with the extra 'costs' complexity

var board[9]:{0..8} = [3,5,6,  // 0,1,2 :index
                       0,2,7,  // 3 4 5 :index
                       8,4,1]; // 6,7,8 :index
hvar empty:{0..8} = 3; //empty position is a secondary variable, no need to put it in the state space

var c = 0; // cost utility, e.g. costs 1 for left and right move, 2 for up, 0 for down

Game() = Left() [] Right() [] Up() [] Down();

Left() = [empty!=2 && empty!=5 && empty!=8] left                    // c=c+1
         {board[empty]=board[empty+1]; board[empty+1]=0; empty=empty+1; c++} -> Game();

Right() = [empty!=0 && empty!=3 && empty!=6] right
          {board[empty]=board[empty-1]; board[empty-1]=0; empty=empty-1; c++} -> Game();

Up() = [empty!=6&&empty!=7&&empty!=8] up
       {board[empty]=board[empty+3]; board[empty+3]=0; empty=empty+3; c=c+2} -> Game();

Down() = [empty!=0&&empty!=1&&empty!=2] down
         {board[empty]=board[empty-3]; board[empty-3]=0; empty=empty-3} -> Game();


#define goal board[0] == 1 && board[1] == 2 && board[2] == 3 &&
             board[3] == 4 && board[4] == 5 && board[5] == 6 &&
             board[6] == 7 && board[7] == 8 && board[8] == 0;

#assert Game() reaches goal with min(c);
```

3	5	6
	2	7
8	4	1

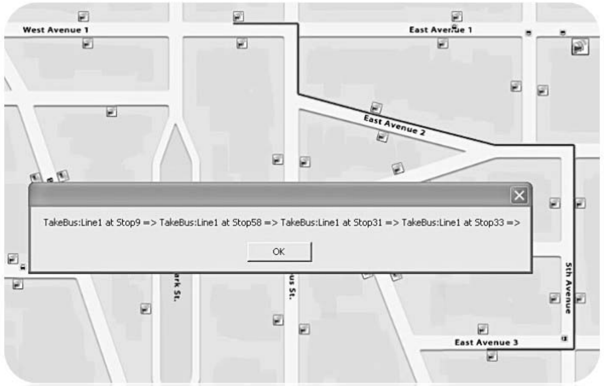
1	2	3
4	5	6
7	8	

Model Checking as Planning/Scheduling/Service:

Transport4You, an intelligent public transportation manager

ICSE 2011 SCORE Competition Project (PAT won FM Award)

- PAT model checker is used not only as a verification tool for the system design but also as a service that computes an optimal travel plan.



- 94 teams from 48 universities in 22 countries started the competition; 55 finished and made final submission; 18 teams were selected for the second round; 5 finalist teams invited to Hawaii with 2000USD travel award for each team. Two winners (Formal Methods Award and Overall Award) were selected during the conference.

  PAT student team won Formal Method Award

Model Checking Concurrent Timed Systems

- A language for modeling compositional real-time systems using implicit clocks.
  - Concurrency + Hierarchy + Data
  - Real-time constructs: wait, within, deadline, timeout …
- A method for abstracting and verifying the models.
  - Zone abstraction
  - Reachability checking, LTL, trace refinement checking and timed refinement checking.

Real-Time Concurrent Processes

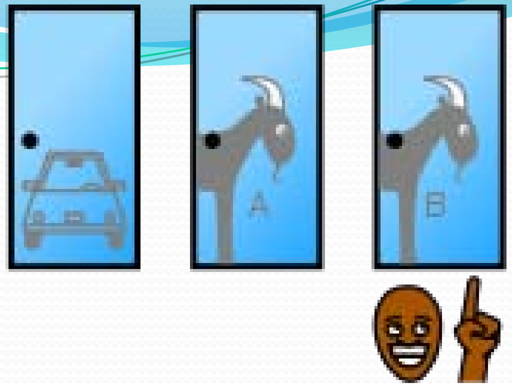
This mutual exclusion protocol is proposed by Fischer in 1985. Mutual exclusion in Fischer's Protocol is guaranteed by carefully placing bounds on the execution times of the instructions, leading to a protocol which is very simple, and relies heavily on time aspects.

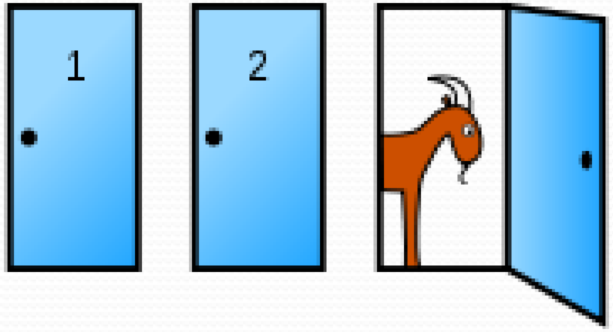
```
#define N 4;
#define Delta 3;
#define Epsilon 4;
#define Idle -1;

var x = Idle;
var counter;

//timed version
P(i) = ifb(x == Idle) {
        ((update.i{x = i} -> Wait[Epsilon]) within[Delta]);
            if (x == i) {
                cs.i{counter++} -> exit.i{counter--; x=Idle} -> P(i)
            } else {
                P(i)
            }
        };

FischersProtocol = ||| i:{0..N-1}@P(i);

//verifying mutual exclusion by reachability analysis
#define MutualExclusionFail counter > 1;
#assert FischersProtocol reaches MutualExclusionFail;
```

Probabilistic Model Checking

- Syntax
  - Hierarchical concurrent systems with probabilistic choices
- Semantics
  - Markov decision processes
- Given a property, probabilistic model checking returns, instead of true or false
  - the maximum and minimum probability of satisfying the property.

Monty Hall Problem



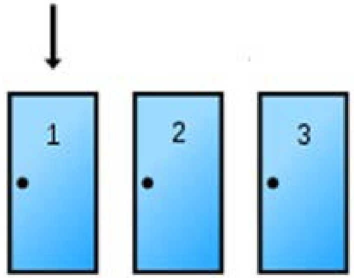


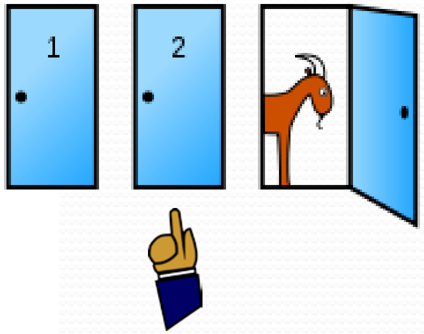
The **Monty Hall problem** is based on the American television game show *Let's Make a Deal* and named after the show's original host, Monty Hall.
The problem was originally posed in a letter by Steve Selvin to the *American Statistician* in 1975.

- In search of a new car, the player picks a door, say 1. The game host then opens one of the other doors, say 3, to reveal a goat and offers to let the player pick door 2 instead of door 1. Should the player take the offer?
- What if the host is dishonest, e.g., place car after 1^{st} guess or host do a switch 33% time after the guess?

PAT Model

```
enum{Door1, Door2, Door3};

var car = -1;
var guess = -1;
var goat = -1;
var final = false;

#define goal guess == car && final;

PlaceCar = []i:{Door1,Door2,Door3}@ placecar.i{car=i} -> Skip;

Guest = pcase {
        1 : guest.Door1{guess=Door1} -> Skip
        1 : guest.Door2{guess=Door2} -> Skip
        1 : guest.Door3{guess=Door3} -> Skip
};

Goat = []i:{Door1,Door2,Door3}@
        ifb (i != car && i != guess) {
                hostopen.i{goat = i} -> Skip
        };

TakeOffer = []i:{Door1,Door2,Door3}@
        ifb (i != guess && i != goat) {
                changeguess{guess = i; final = true} -> Stop
        };

NotTakeOffer = keepguess{final = true} -> Stop;

Sys_Take_Offer = PlaceCar; Guest; Goat; TakeOffer;

#assert Sys_Take_Offer reaches goal with prob;

Sys_Not_Take_Offer = PlaceCar; Guest; Goat; NotTakeOffer;

#assert Sys_Not_Take_Offer reaches goal with prob;
```





What if the host is Dishonest?

```
//place after guessing
Sys_With_Dishonest_Program = Guest; PlaceCar; Goat; NotTakeOffer;

#assert Sys_With_Dishonest_Program reaches goal with prob;

HostSwitch = pcase {
                1 : switch{car = guess} -> Skip
                2 : Skip
        };

Sys_With_Cheating_Host_Switch = PlaceCar; Guest; Goat; HostSwitch; TakeOffer;

#assert Sys_With_Cheating_Host_Switch reaches goal with prob;

Sys_With_Cheating_Host_Not_Switch = PlaceCar; Guest; Goat; HostSwitch; NotTakeOffer;

#assert Sys_With_Cheating_Host_Not_Switch reaches goal with prob;
```

Combine Real-Time and Probability

Passing me without stopping!

```
#import "PAT.Lib.Lift";
#define NoOfFloors 2;
#define NoOfLifts 2;
var<LiftControl> ctrl = new LiftControl(NoOfFloors,NoOfLifts);
var passby = 0;

aSystem = (||| x:{0..NoOfLifts-1} @ Lift(x, 0, 1)) ||| Requests();

Requests() = Request();Request();
Request() = pcase {
            1 : extreq.0.1{ctrl.AssignExternalRequest(0,1)} -> Skip
            1 : intreq.0.0.1{ctrl.AddInternalRequest(0,0)} -> Skip
            1 : intreq.1.0.1{ctrl.AddInternalRequest(1,0)} -> Skip
            1 : extreq.1.0{ctrl.AssignExternalRequest(1,0)} -> Skip
            1 : intreq.0.1.1{ctrl.AddInternalRequest(0,1)} -> Skip
            1 : intreq.1.1.1{ctrl.AddInternalRequest(1,1)} -> Skip
       } within[1];

Lift(i, level, direction) = case {
            ctrl.isToOpenDoor(i, level) == 1 : (serve.level.direction{ctrl.ClearRequests(i, level, direction)}
                                                -> Lift(i, level, direction))
            ctrl.KeepMoving(i, level, direction) == 1 : (reach.level+direction.direction
                                                         {passby = ctrl.UpdateLiftStatus(i, level, direction)}
                                                         -> Lift(i, level+direction, direction))
            ctrl.HasAssignment(i) == 1 : changedirection.i{ctrl.ChangeDirection(i)}
                                         -> Lift(i, level, -1*direction)
            default : idle.i -> Lift(i, level, direction)
       } within[2];

#define goal passby == 1;
#assert aSystem reaches goal with prob;
```

The Current Status

- PAT is available at http://pat.comp.nus.edu.sg
- 1Million lines of code, 11 modules with 100+ build in examples
- Used as an educational tool in many universities.
- Attracted more than 1700 registered users in the last 3 years from more than 350 organizations, e.g. Microsoft, HP, ST Elec, Oxford Univ., … Sony, Hitachi, Canon.
- Japanese PAT User group formed in Sep 2009:

Founding Members:
Hiroshi Fujimoto
Nobukazu Yoshioka
Toshiyuki Fujikura
Kenji Taguchi
Masaru Nagaku
Kazuto MATSUI

Some related and background papers

- Jun Sun, Yang Liu, Jin Song Dong, Yan Liu, Ling Shi, Etienne, Andre. **Modeling and Verifying Hierarchical Real-time Systems using Stateful Timed CSP**. The ACM Transactions on Software Engineering and Methodology (TOSEM). (Accepted)
- Yang Liu, Jun Sun and Jin Song Dong. **PAT 3: An Extensible Architecture for Building Multi-domain Model Checkers**. The 22nd annual International Symposium on Software Reliability Engineering (ISSRE 2011), Hiroshima, Japan, Nov 29 - Dec 2, 2011.
- Jun Sun, Yang Liu, Songzheng Song and Jin Song Dong. **PRTS: An Approach for Model Checking Probabilistic Real-time Hierarchical Systems**. The 13th International Conference on Formal Engineering Methods (ICFEM 2011), pages 147-162, Durham, United Kingdom, October 25-28, 2011.
- Shaojie Zhang, Jun Sun, Jun Pang, Yang Liu and Jin Song Dong. **On Combining State Space Reductions with Global Fairness Assumptions**. The 17th International Symposium on Formal Methods (FM 2011), pages 432 - 447, Lero, Limerick, Ireland, June 20 - 24, 2011.
- Y. Liu, J. Sun and J. S. Dong. **Analyzing Hierarchical Complex Real-time Systems**. *The ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE 2010)*, Santa Fe, New Mexico, USA, 7-11 November 2010.
- C. Chen, J. S. Dong, J. Sun and A. Martin. **A Verification System for Interval-based Specification Languages,** ACM Transactions on Software Engineering and Methodology, Volume 19(4), pages 1 - 36, ACM. 2010
- C. Chen, J. S. Dong and J. Sun. **A Formal Framework for Modeling and Validating Simulink Diagrams.** Formal Aspects of Computing. 21(5), pages 451-483, Springer. Oct, 2009.
- J. Sun, Y. Liu, J. S. Dong and J. Pang. **PAT: Towards Flexible Verification under Fairness.** The 21th International Conference on Computer Aided Verification (CAV 2009), Grenoble, France, June 2009.
- J. S. Dong, P. Hao, S. C. Qin, J. Sun and Y. Wang, **Timed Automata Patterns**. IEEE Transactions on Software Engineering, vol. 34(6), pp 844-859, Nov./Dec. 2008.
- C. Chen, J. S. Dong and J. Sun. **A Verification System for Timed Interval Calculus**, the 30th International Conference on Software Engineering (ICSE'08), 2008.
- K. Taguchi and J. S. Dong. **Formally Specifying and Verifying Mobile Agents : Model Checking Mobility,** International Journal of Agent-Oriented Software Engineering, 2008.
- J. Sun and J. S. Dong, **Design Synthesis from Interaction and State-based Specifications.** IEEE Transactions on Software Engineering, Vol-32(6):349-364, 2006
- H. Wang, J. S. Dong, J. Sun and J. Sun. **Reasoning Support for Semantic Web Ontology Family Languages Using Alloy.** International Journal of Multiagent and Grid Systems, IOS press, Vol-2(4):455-471, 2006.
- L. Yuan, J. S. Dong, J. Sun and H. A. Basit. **Generic Fault Tolerant Software Architecture Reasoning and Customization.** IEEE Transactions on Reliability. Vol-55(3):421-435, 2006
- J. Sun, J. S. Dong, S. Jarzabek and H. Wang. **CAD System Family Architecture and Verification: An Integrated Formal Approach.** IEE Proceedings Software. Vol-153(3):102-112, 2006.
- J. Sun and J. S. Dong, **Synthesis of Distributed Processes from Scenario-based Specifications.** Formal Methods 2005 (FM'05), LNCS, pp 415-431, Newcastle, UK. 2005.
- J.S. Dong and P. Hao and B. Mahony, **Formal Designs for Embedded and Hybrid Systems.** International Journal on Software Engineering and Knowledge Engineering, vol 15(2): 373-378, 2005
- X. Wang, J. S. Dong, C. Chin, S. R. Hettiarachchi and D. Zhang. **Semantic Space: A Semantic Web Infrastructure for Smart Spaces.** IEEE Pervasive Computing, 3(3):32-39, July-September 2004.
- S. C. Qin, J. S. Dong and W. N. Chin. **A Semantic Foundation of TCOZ in Unifying Theory of Programming.** FM'03. LNCS, pages 321-340, 2003.
- B. Mahony and J.S. Dong. **Deep Semantic Links of TCSP and Object-Z: TCOZ Approach**. Formal Aspects of Computing journal, 13:142-160, Springer, 2002.
- B. Mahony and J.S. Dong. **Timed Communicating Object Z.** IEEE Transactions on Software Engineering, 26(2):150-177, Feb 2000.
- B. Mahony and J.S. Dong. **Sensors and Actuators in TCOZ**. World Congress on Formal Methods (FM'99), LNCS, Vol 1709, pages 1166-1185, Springer-Verlag, Toulouse, 1999.
- B. Mahony and J.S. Dong. **Blending Object-Z and Timed CSP: an introduction to TCOZ**. In Proceedings of the 20th International Conference on Software Engineering (ICSE'98), editors: K. Futatsugi and R. Kemmerer and K. Torii, IEEE Press, pages 95-104, Kyoto, 1998.