Formal Dependability Analysis using Theorem Proving

Waqar Ahmed

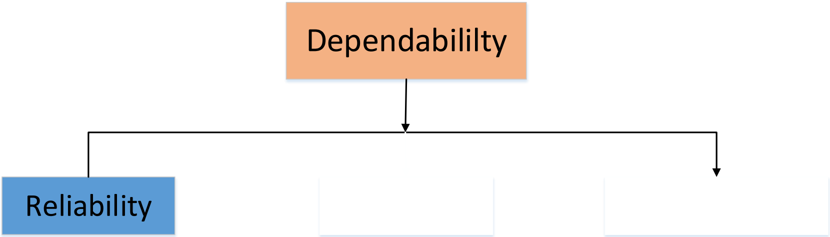
ACL2 Seminar

University of Texas at Austin, Tx, USA

January 20, 2017

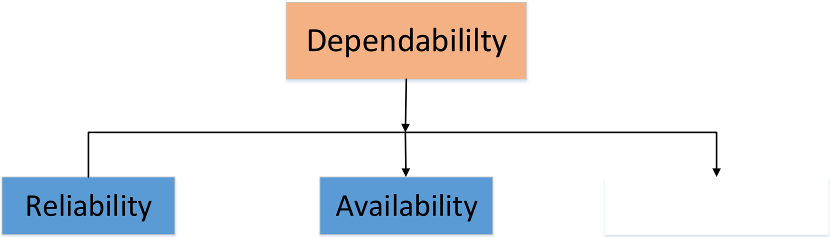
Outline

Dependability



The ability of system to deliver services as specified within a given time

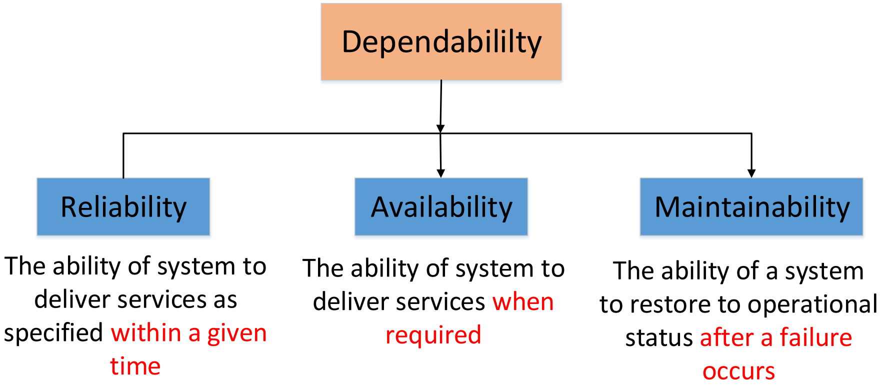
Dependability



The ability of system to deliver services as specified within a given time

The ability of system to deliver services when required

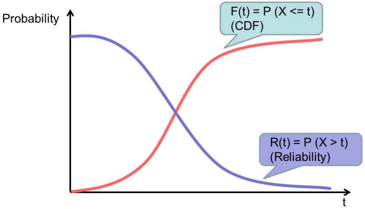
Dependability



Formal Definitions

- Reliability $= \mathbb{P}$(no failure occurs before certain time)

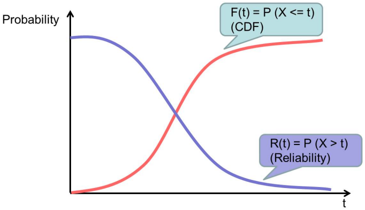
$$\begin{aligned} R(t) &= Pr(X > t) \\ &= 1 - Pr(X \leq t) \\ &= 1 - F_X(t) \end{aligned}$$



Formal Definitions

- Reliability $= \mathbb{P}$(no failure occurs before certain time)

$$\begin{aligned} R(t) &= Pr(X > t) \\ &= 1 - Pr(X \leq t) \\ &= 1 - F_X(t) \end{aligned}$$

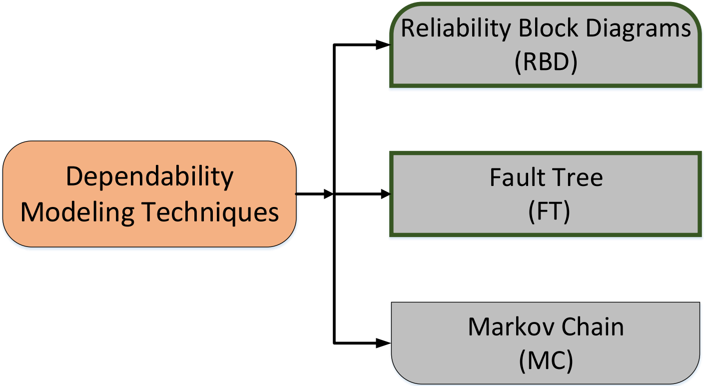


- Availability is typically derived from reliability and maintainability measures
  - $A(t) = \dfrac{MTBF}{MTBF + MTTR}$

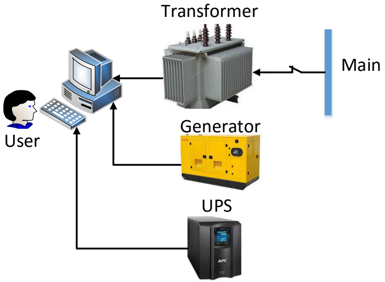
    where MTBF = MTTF + MTTR
    - MTBF = Mean time between failures (Reliability Metric)
    - MTTF = Mean time to failure (Reliability Metric)
    - MTTR = Mean time to repair (Maintainability Metric)

Outline

Dependability Modeling Techniques



Example: Power Supply System

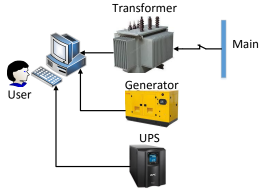


- User requires continuous supply of power for his Lab PC
  - The UPS can support the load during a switch from the main supply to the generator
- Wants to determine the reliability of power supply system

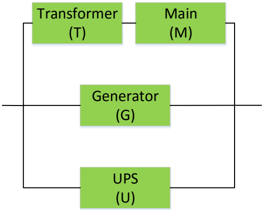
Example: Power Supply System

Step 1

Construct an RBD Model



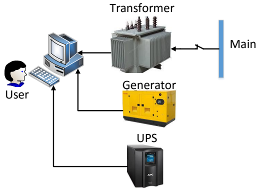
Power Supply RBD



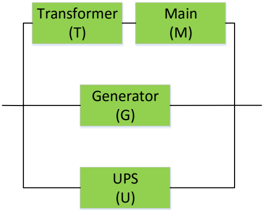
Example: Power Supply System

Step 1

Construct an RBD Model



Power Supply RBD



$$\mathtt{pow\_sys\_rbd} = (\mathtt{M} \cap \mathtt{T}) \cup \mathtt{G} \cup \mathtt{U}$$

Example: Power Supply System

Step 2

Identify the RBD type

Step 3

Assigning failure distribution to each system components, i.e., $e^{-\lambda t}$

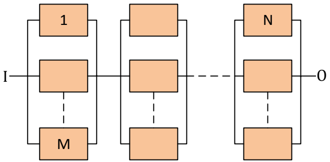
Step 3

Use the corresponding mathematical expression to evaluate the overall reliability based on the sub-components reliability

$$\mathbb{P}((\mathtt{M} \cap \mathtt{T}) \cup \mathtt{G} \cup \mathtt{U}) = 1 - (1 - \mathbb{P}(\mathtt{M}) * \mathbb{P}(\mathtt{T})) * (1 - \mathbb{P}(\mathtt{G})) * (1 - \mathbb{P}(\mathtt{U}))$$
$$= 1 - (1 - \mathtt{e}^{\mathtt{Mt}} * \mathtt{e}^{-\mathtt{Tt}}) * (1 - \mathtt{e}^{-\mathtt{Gt}}) * (1 - \mathtt{e}^{-\mathtt{Ut}})$$

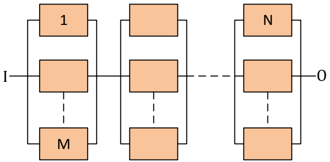
Reliability Block Diagrams

- Model the failure relationship of system components as a diagram of sub-blocks and connectors (RBD)
- Judge the failure characteristics of the overall system based on the failure rates of sub-blocks



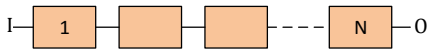
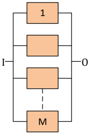
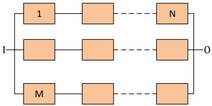
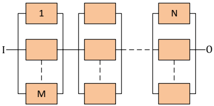
Reliability Block Diagrams

- Model the failure relationship of system components as a diagram of sub-blocks and connectors (RBD)
- Judge the failure characteristics of the overall system based on the failure rates of sub-blocks

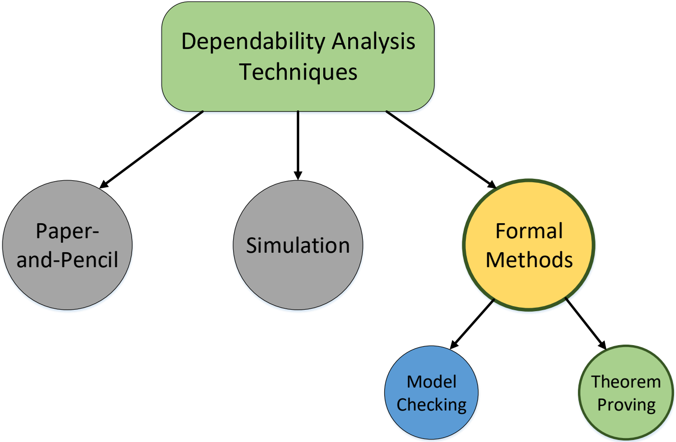


- The overall system failure happens if all the paths for successful execution fail
  - Add more parallelism to meet the dependability goals

Types of RBD

RBDs	Mathematical Expressions
	$R_{series}(t) = Pr(\bigcap_{i=1}^{N} E_i(t)) = \prod_{i=1}^{N} R_i(t)$
	$R_{parallel}(t) = Pr(\bigcup_{i=1}^{M} E_i) = 1 - \prod_{i=1}^{M}(1 - R_i(t))$
	$R_{parallel-series}(t)= Pr(\bigcup_{i=1}^{M} \bigcap_{j=1}^{N} E_{ij}(t))= 1 - \prod_{i=1}^{M}(1 - \prod_{j=1}^{N}(R_{ij}(t)))$
	$R_{series-parallel}(t)= Pr(\bigcap_{i=1}^{N} \bigcup_{j=1}^{M} E_{ij}(t))= \prod_{i=1}^{N}(1 - \prod_{j=1}^{M}(1 - R_{ij}(t)))$

Dependability Analysis Techniques



Comparison

Feature	Paper-and-pencil Proof	Simulation Tools	Model Checking	Higher-order-Logic Theorem Proving

Comparison

Feature	Paper-and-pencil Proof	Simulation Tools	Model Checking	Higher-order-Logic Theorem Proving
Models	Paper (Random Variables)			
Analysis	Analytically (probability distributions, Expressions for RBDs and FTs and MCs)			
Expressiveness	✓ (?)			
Accuracy	✓ (?)			
Automation				

Comparison

Feature	Paper-and-pencil Proof	Simulation Tools	Model Checking	Higher-order-Logic Theorem Proving
Models	Paper (Random Variables)	Computer Program (Pseudo Random Numbers)		
Analysis	Analytically (probability distributions, Expressions for RBDs and FTs and MCs)	Numerical Methods		
Expressiveness	✓ (?)	✓		
Accuracy	✓ (?)			
Automation		✓		

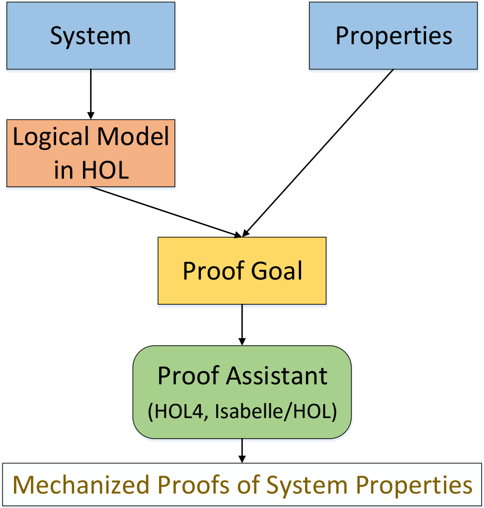
Comparison

Feature	Paper-and-pencil Proof	Simulation Tools	Model Checking	Higher-order-Logic Theorem Proving
Models	Paper (Random Variables)	Computer Program (Pseudo Random Numbers)	State Transition Graph (Markov Chains)	
Analysis	Analytically (probability distributions, Expressions for RBDs and FTs and MCs)	Numerical Methods	State Exploration	
Expressiveness	✓ (?)	✓		
Accuracy	✓ (?)		✓	
Automation		✓	✓	

Comparison

Feature	Paper-and-pencil Proof	Simulation Tools	Model Checking	Higher-order-Logic Theorem Proving
Models	Paper (Random Variables)	Computer Program (Pseudo Random Numbers)	State Transition Graph (Markov Chains)	Logical Function
Analysis	Analytically (probability distributions, Expressions for RBDs and FTs and MCs)	Numerical Methods	State Exploration	Formal Reasoning
Expressiveness	✓ (?)	✓		✓
Accuracy	✓ (?)		✓	✓
Automation		✓	✓	

Higher-order-Logic Theorem Proving



HOL4 Theorem Prover

- Developed at University of Cambridge
- Language: Standard ML
- Logic: Higher-order Logic
- 5 axioms and 8 interference rules

UNIVERSITY OF CAMBRIDGE

Outline

Formalization of RBDs

- Defined new datatype in HOL to model RBDs

Datatype for RBD

```
Hol_datatype‘ rbd = series of rbd list| parallel of rbd list|
            atomic of ’a event ‘
```

Formalization of RBDs

- Defined new datatype in HOL to model RBDs

Datatype for RBD

```
Hol_datatype‘ rbd = series of rbd list| parallel of rbd list|
          atomic of ’a event ‘
```

Definition

```
 (∀ p.  rbd_struct p (series []) = p_space p) ∧
(∀ xs x p.  rbd_struct p (series (x::xs)) =
        rbd_struct p x ∩ rbd_struct p (series xs)) ∧
(∀ p.  rbd_struct p (parallel []) = {}) ∧
(∀ xs x p.  rbd_struct p (parallel (x::xs)) =
        rbd_struct p x ∪ rbd_struct p (parallel xs)) ∧
(∀ p a.  rbd_struct p (atomic a) = a)
```

Formalization of RBDs

- Defined new datatype in HOL to model RBDs

Datatype for RBD

```
Hol_datatype‘ rbd = series of rbd list| parallel of rbd list|
          atomic of ’a event ‘
```

Definition

```
 (∀ p.  rbd_struct p (series []) = p_space p) ∧
(∀ xs x p.  rbd_struct p (series (x::xs)) =
        rbd_struct p x ∩ rbd_struct p (series xs)) ∧
(∀ p.  rbd_struct p (parallel []) = {}) ∧
(∀ xs x p.  rbd_struct p (parallel (x::xs)) =
        rbd_struct p x ∪ rbd_struct p (parallel xs)) ∧
(∀ p a.  rbd_struct p (atomic a) = a)
```

Formalization of RBDs

- Defined new datatype in HOL to model RBDs

Datatype for RBD

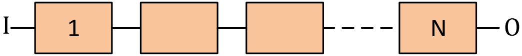
```
Hol_datatype‘ rbd = series of rbd list| parallel of rbd list|
          atomic of ’a event ‘
```

Definition

```
 (∀ p.  rbd_struct p (series []) = p_space p) ∧
(∀ xs x p.  rbd_struct p (series (x::xs)) =
        rbd_struct p x ∩ rbd_struct p (series xs)) ∧
(∀ p.  rbd_struct p (parallel []) = {}) ∧
(∀ xs x p.  rbd_struct p (parallel (x::xs)) =
        rbd_struct p x ∪ rbd_struct p (parallel xs)) ∧
(∀ p a.  rbd_struct p (atomic a) = a)
```

Series RBD

- All components should be functional for the system to be functional



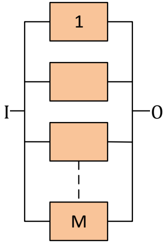
$$R_{series}(t) = Pr(\bigcap_{i=1}^{N} E_i(t)) = \prod_{i=1}^{N} R_i(t)$$

HOL Formalization

```
⊢ ∀ p L. prob_space p ∧
(∀x'.  MEM x' L ⇒ x' ∈ events p) ∧ ¬ NULL L ∧
mutual_indep p L ⇒
 (prob p (rbd_struct p (series (rbd_list L))) =
  list_prod (list_prob p L))
```

Parallel RBD

- At least one components should be functional

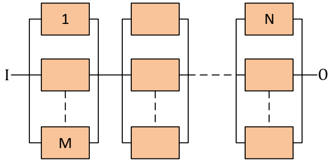


$$R_{parallel}(t) = Pr(\bigcup_{i=1}^{M} E_i) = 1 - \prod_{i=1}^{M}(1 - R_i(t))$$

HOL Formalization

```
⊢ ∀ p L. prob_space p ∧
(∀x'. MEM x' L ⇒ x' ∈ events p) ∧ ¬ NULL L ∧
mutual_indep p L ⇒
 (prob p (rbd_struct p (parallel (rbd_list L))) =
  1 - list_prod (one_minus_list (list_prob p L)))
```

Series-Parallel RBD



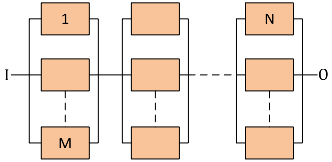
$$R_{series-parallel}(t) = Pr(\bigcap_{i=1}^{N} \bigcup_{j=1}^{M} E_{ij}(t))$$

$$= \prod_{i=1}^{N} (1 - \prod_{j=1}^{M} (1 - R_{ij}(t)))$$

HOL Formalization

⊢ ∀ p L. prob_space p ∧
(∀z. MEM z L ⇒ ¬NULL z) ∧
(∀x'. MEM x' (FLAT L) ⇒ x' ∈ events p) ∧
mutual_indep p (FLAT L) ⇒
(prob p (rbd_struct p
((series of (λa. parallel (rbd_list a))) L)) =
(list_prod of
(λa. 1 - list_prod (one_minus_list (list_prob p a)))) L)

Series-Parallel RBD



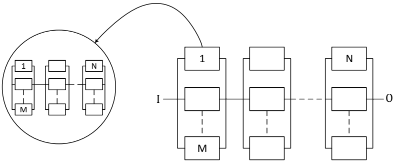
$$R_{series-parallel}(t) = Pr(\bigcap_{i=1}^{N}\bigcup_{j=1}^{M} E_{ij}(t))$$

$$= \prod_{i=1}^{N}(1 - \prod_{j=1}^{M}(1 - R_{ij}(t)))$$

HOL Formalization

```
⊢ ∀ p L. prob_space p ∧
(∀z. MEM z L ⇒ ¬NULL z) ∧
(∀x'. MEM x' (FLAT L) ⇒ x' ∈ events p) ∧
 mutual_indep p (FLAT L) ⇒
(prob p (rbd_struct p
    ((series of (λa. parallel (rbd_list a))) L)) =
 (list_prod of
  (λa. 1 - list_prod (one_minus_list (list_prob p a)))) L)
```

Nested Series-Parallel RBD



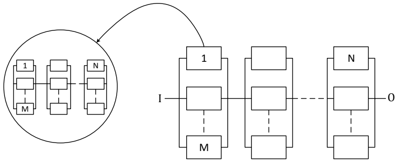
$$R(t) = Pr(\bigcap_{i=1}^{N}\bigcup_{j=1}^{M}(\bigcap_{k=1}^{N}\bigcup_{l=1}^{M} A_{ijkl}(t)))$$

$$= \prod_{i=1}^{N}(1 - \prod_{j=1}^{M}(1 - (\prod_{k=1}^{N}(1 - \prod_{l=1}^{M}(1 - R_{ijkl}(t)))))$$

HOL Formalization

⊢ ∀ p L. prob_space p ∧ (∀ z. MEM z (FLAT (FLAT L)) ⇒ ¬NULL z) ∧
(∀ x'. MEM x' (FLAT (FLAT (FLAT L))) ⇒ x' ∈ events p) ∧
mutual_indep p (FLAT (FLAT (FLAT L))) ⇒
(prob p (rbd_struct p ((series of parallel of series of
(λa. parallel (rbd_list a))) L)) =
(list_prod of (λa. 1 - list_prod (one_minus_list a)) of
(λa. list_prod a) of
(λa. 1 - list_prod (one_minus_list (list_prob p a)))) L)

Nested Series-Parallel RBD



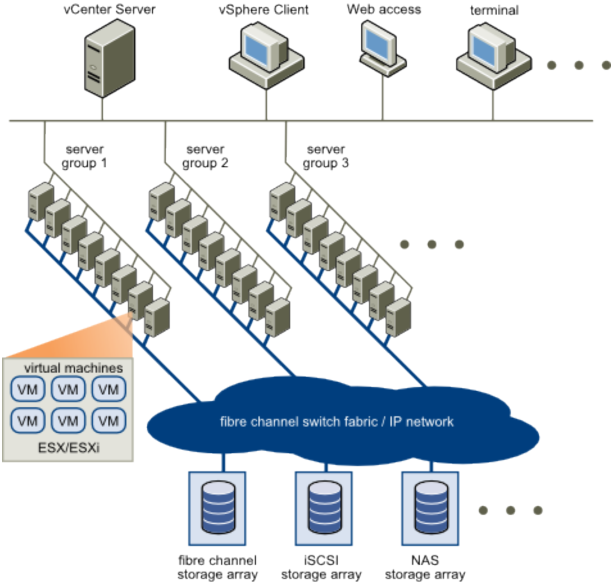
$$R(t) = Pr(\bigcap_{i=1}^{N}\bigcup_{j=1}^{M}(\bigcap_{k=1}^{N}\bigcup_{l=1}^{M} A_{ijkl}(t)))$$

$$= \prod_{i=1}^{N}(1 - \prod_{j=1}^{M}(1 - (\prod_{k=1}^{N}(1 - \prod_{l=1}^{M}(1 - R_{ijkl}(t)))))$$

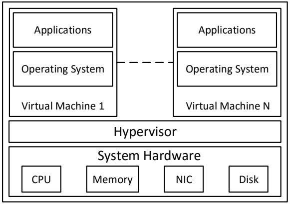
HOL Formalization

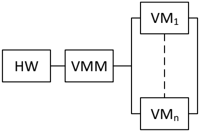
⊢ ∀ p L. prob_space p ∧ (∀ z. MEM z (FLAT (FLAT L)) ⇒
¬NULL z) ∧
(∀ x'. MEM x' (FLAT (FLAT (FLAT L))) ⇒ x' ∈ events p) ∧
mutual_indep p (FLAT (FLAT (FLAT L))) ⇒
(prob p (rbd_struct p ((series of parallel of series of
(λa. parallel (rbd_list a))) L)) =
(list_prod of (λa. 1 - list_prod (one_minus_list a)) of
(λa. list_prod a) of
(λa. 1 - list_prod (one_minus_list (list_prob p a)))) L)

Application: Virtual Data Center



Cloud Server



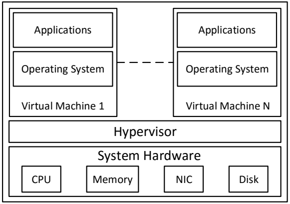


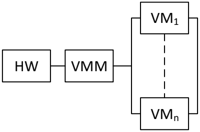
$$R_{Server} = (\exp^{-(\lambda_{VMM}+\lambda_{HW})t})[1 - \prod_{i=1}^{n}(1 - \exp^{-\lambda_{VM_i} t})]$$

HOL Formalization

⊢ ∀ X_VM X_VMM X_HW C_VM C_VMM C_HW p t.
¬NULL X_VM ∧ 0 ≤ t ∧ prob_space p ∧
in_events p (rel_event_list p ([[X_VMM];[X_HW];X_VM]) t) ∧
mutual_indep p (rel_event_list p ([[X_VMM];[X_HW];X_VM]) t) ∧
LENGTH C_VM = LENGTH X_VM ∧
exp_dist_list p [[X_VMM];[X_HW];X_VM] [[C_VMM];[C_HW];C_VM] ⇒
(prob p (rbd_virt_cloud_server p X_VMM X_HW X_VM t =
exp (-(C_VMM + C_HW) * t) *
(1 - list_prod (one_minus_list (exp_func_list C_VM t)))

Cloud Server



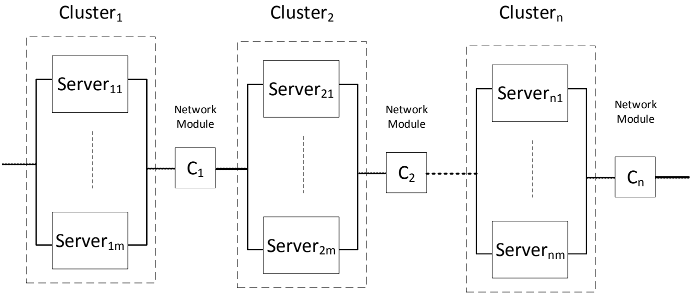


$$R_{Server} = (\exp^{-(\lambda_{VMM}+\lambda_{HW})t})[1 - \prod_{i=1}^{n}(1 - \exp^{-\lambda_{VM_i} t})]$$

HOL Formalization

```
⊢ ∀ X_VM X_VMM X_HW C_VM C_VMM C_HW p t.
¬NULL X_VM ∧ 0 ≤ t ∧ prob_space p ∧
in_events p (rel_event_list p ([[X_VMM];[X_HW];X_VM]) t) ∧
mutual_indep p (rel_event_list p ([[X_VMM];[X_HW];X_VM]) t) ∧
LENGTH C_VM = LENGTH X_VM ∧
exp_dist_list p [[X_VMM];[X_HW];X_VM] [[C_VMM];[C_HW];C_VM] ⇒
  (prob p (rbd_virt_cloud_server p X_VMM X_HW X_VM t =
  exp (-(C_VMM + C_HW) * t) *
    (1 - list_prod (one_minus_list (exp_func_list C_VM t)))
```

VDC RBD



$$R_{VDC_{nm}} = \prod_{i=1}^{n} [1 - \prod_{j=1}^{m} (1 - R_{Server_{ij}}) * \exp^{-\lambda_{C_i} t}]$$

Reliability of Virtual Data Center

```
Theorem 8:  ⊢ ∀ X_VM X_VMM X_HW X_C C_VM C_VMM C_HW C m n p t.
[A1]:  0 ≤ t ∧ prob_space p ∧
[A2]:  ¬NULL (cloud_server_rv_list [X_VM] m n) ∧ ¬NULL X_VM ∧
       ¬NULL (cloud_server_fail_rate_list [C_VM] m n) ∧ ¬NULL C_VM ∧
[A3]:  not_null_list
        (FLAT (FLAT (cloud_server_rv_list [X_VM] m n))) ∧
      ¬NULL (rel_event_list p X_C t) ∧
[A4]:  (LENGTH C = LENGTH X_C) ∧ (LENGTH X_VM = LENGTH C_VM) ∧
[A5]:  in_events p (FLAT (FLAT (FLAT (four_dim_rel_event_list p
         (cloud_server_rv_list [X_VM] m n) t)))) ∧
[A6]:  rel_event p X_VMM t ∈ events p ∧
       rel_event p X_VM t ∈ events p ∧
       rel_event p X_HW t ∈ events p ∧
       in_events p (rel_event_list p X_C t) ∧
[A7]:  exp_dist_list p X_C C ∧
       four_dim_exp_dist_list p
         (cloud_server_rv_list [[X_VMM];[X_HW];X_VM] m n)
         (cloud_server_fail_rate_list [[C_VMM];[C_HW];C_VM] m n) ∧
[A8]:  mutual_indep p (rel_event_list p X_C t ++
        FLAT (FLAT (FLAT (four_dim_rel_event_list p
         (cloud_server_rv_list [[X_VMM];[X_HW];X_VM] m n) t)))) ⇒
  (prob p (rbd_VDC_cloud p X_C X_VMM X_HW X_VM m n t) =
   list_prod (exp_func_list C t) *
     (list_prod of (λa.  1 - list_prod (one_minus_list a)) of
      (λa.  list_prod a) of
       (λa.  1 - list_prod (one_minus_list (exp_func_list a t))))
        (cloud_server_fail_rate_list [[C_VMM];[C_HW];C_VM] m n))
```

Dependability Computation

Amazon Data Centers	# of Server Racks	# of Servers
US East (Virginia)	5,030	321,920
US West (Oregon)	41	2,624
US West (N. California)	630	40,320
EU West (Ireland)	814	52,096
SA East (Sao Paulo)	25	1600

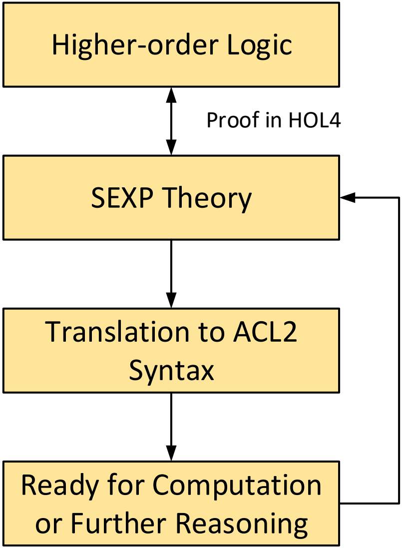
- Translate HOL exponential expression to ML
  - Slower
- HOL4/ACL2 Link
  - Fast Lisp
  - Highly automatic features for reasoning

Outline

History

- 1991 - Proof Manager Tool by Fink et. al
  - Translates HOL input to first order assertions for Boyer-Moor prover
- 1999 - ACL2PII by Mark Staples
  - Linking HOL to ACL2 at ML level
  - No reasoning capabilities
- 2005 - HOL4/ACL2 link
  - Formal model of ACL2s (sexp Theory) intended universe in HOL
  - Deductive Reasoning
  - Ability to port functions either way
    - HOL4 -> ACL2
    - ACL2 -> HOL4

Flow Between HOL and ACL2



Porting HOL Exponential Function

HOL Definition

```
⊢ ∀ n a.
 exp_ratr a n =
 if n = 0 then 1
 else if 0 < n then
   a * exp_ratr a (n - 1)
 else rat_minv a * exp_ratr a (n + 1)
```

SEXP Definition

```
⊢ acl2_expt a n =
if zip n = nil then
  ite (equal (fix a) (int 0)) (int 0)
   (if less (int 0) n = nil then
      mult (reciprocal a) (acl2_expt a (add n (int 1)))
    else mult a (acl2_expt a (add n (unary_minus (int 1)))))
 else int 1
```

Proving Equivalence

Theorem

$\vdash \forall$b a. a $\neq$ 0 $\Rightarrow$

(rat (exp_ratr a b) = acl2_expt (rat a) (int b))

Translating to ACL2 Syntax

- Automatic translator available in the existing Link

```
fun pr_sexp t = pr_mlsexp(term_to_mlsexp t)
```

```
pr_sexp `` mult ((acl2_expt (cpx 10 27 0 1) (int (-&2))))
(add (add (acl2_expt (cpx 10 27 0 1) (int (-&3)))
          (acl2_expt (cpx 10 27 0 1) (int (-&7))))
     (acl2_expt (cpx 10 27 0 1) (int (-&5))))`` ;
```

```
(ACL2::BINARY-* (ACL2::EXPT 10/27 -2/1)
(ACL2::BINARY-+ (ACL2::BINARY-+ (ACL2::EXPT 10/27 -3/1)
                                (ACL2::EXPT 10/27 -7/1))
                (ACL2::EXPT 10/27 -5/1)))
```

ACL2 Output

8815121875287/1000000000

Maclaurin Series

$$f(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 \cdots$$

- Represents a function as sum of terms
- Better approximation depends upon of number of terms in a series
- Negative exponential produces alternating series

  $e^{-x} = \sum_{m=0}^{n}(-1)^m \frac{x^{(m+1)}}{(m+1)!}$

Outline

Error Bound Property

$$|S(0,n) - S(0,m)| <= a_{(m+1)+1}$$

where $S(m,n) = \sum_m^n (-1)^m a_m$

- Series must be convergent
- Each term should be positive
- Proof Approach: Split into two cases
  - $|S(0,2n) - S(0,m)| <= a_{(m+1)+1}$
  - $|S(0,2n+1) - S(0,m)| <= a_{(m+1)+1}$

ACL2 Proof Approach

- Use constraint function

```
(encapsulate
(((n-term *) => *)) ; (n-term n) is the |nth term| in our series
(local (defun n-term (n)
    (/ (+ 1 n))))
(defthm positive-rationalp-n-term
  (implies (natp n)
    (and (rationalp (n-term n))
         (< 0 (n-term n))))
:rule-classes :type-prescription)
(defthm n-term-decreases
  (implies (and (natp n)
                (<= 0 n))
           (< (n-term (+ n 1))
              (n-term n)))
:rule-classes :linear))
```

Even and Odd Terms

$|S(0, 2n) - S(0, m)| <= a_{(m+1)+1}$

```
(defthm abs-n-term-sum-even-le-n-term
(implies (and (natp m)
              (natp n))
         (<= (abs(- (n-term-sum 0 (+ m 1 (* 2 n)))
                    (n-term-sum 0 m)))
             (n-term (+ (+ m 1) 1))))
```

$|S(0, 2n) - S(0, m)| <= a_{(m+1)+1}$

```
(defthm abs-n-term-sum-odd-le-n-term
(implies (and (natp m)
              (natp n))
         (<= (abs(- (n-term-sum 0 (+ m 1 (* 2 n) 1))
                    (n-term-sum 0 m)))
             (n-term (+ (+ m 1) 1))))
```

Alternating Series Error Bound Property

$|S(0,n) - S(0,m)| <= a_{(m+1)+1}$

```
(defthm abs-n-term-sum-le-n-term
(implies (and (natp m)
              (natp n))
         (<= (abs(- (n-term-sum 0 (+ m 1 n))
                    (n-term-sum 0 m)))
             (n-term (+ (+ m 1) 1)))))
```

Error Bound Property

$$|exp(0,n) - exp(0,m)| \leq \frac{x^{(m+1)+1}}{((m+1)+1)!}$$

- Using Functional instantiation

ACL2 Formalization

```
(defthm abs-expt-error-bound
(implies (and (rationalp x)
              (< 0 x)
              (natp k)
              (natp n)
              (< x (+ k 1)))
        (<= (abs (- (expt-minus-maclaurin x 0 (+ k 1 n))
                    (expt-minus-maclaurin x 0 k)))
            (expt-div-fact x (+ (+ k 1) 1)))))
```

Outline

Conclusion

- Dependability
    - Reliability
    - Availability
    - Maintainability

Conclusion

- Dependability
  - Reliability
  - Availability
  - Maintainability
- Dependability Modeling Techniques
  - Reliability Block Diagram
  - Fault Tree
  - Markov Chains
- Formal Dependability Analysis Techniques
  - Model Checkng
  - Interactive Theorem Proving

Conclusion

- Dependability
    - Reliability
    - Availability
    - Maintainability
- Dependability Modeling Techniques
    - Reliability Block Diagram
    - Fault Tree
    - Markov Chains
- Formal Dependability Analysis Techniques
    - Model Checkng
    - Interactive Theorem Proving
- Benefits
    - Reason about key dependability properties of the system
    - Computational capabilities using HOL4/ACL2 Link

Thanks!

save.seecs.nust.edu.pk