



KTH Computer Science
and Communication

Learning-based Testing for Procedural and Hybrid Systems

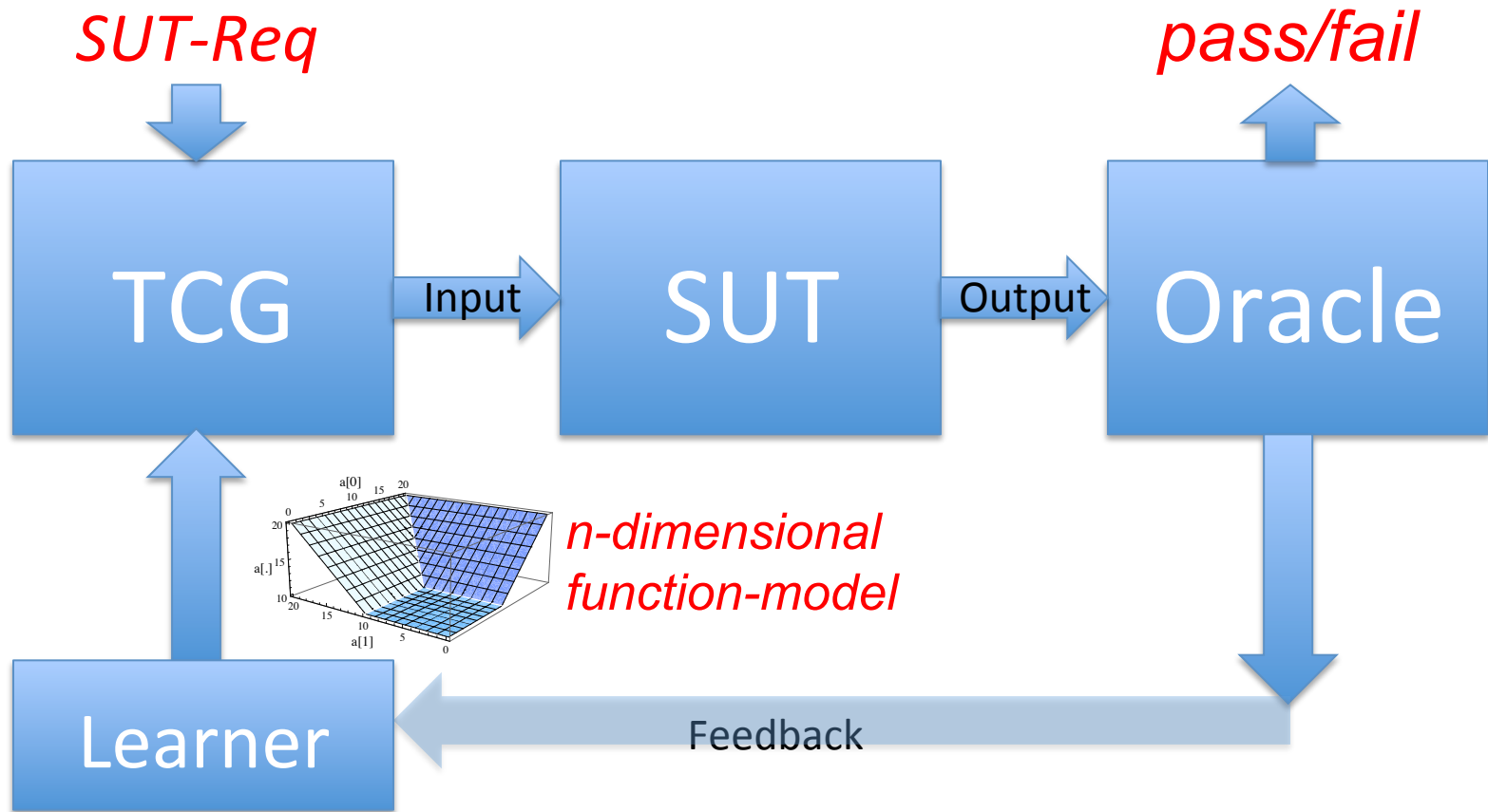
Karl Meinke,
karlm@kth.se

KTH Royal Institute of Technology Stockholm

0. Overview of Talk

1. Introduction and Motivation
2. Technical Approach
3. Benchmarking Results
4. Conclusions

1.2. LBT for Procedural Code



1.3. LBT for Numerical Analysis

- Requirements Language – Hoare logic over
 - first-order logic over real-closed fields
- Models
 - non-gridded n-dimensional piecewise polynomials of low degree, $n=1,2,3$ (= “n-wise testing”)
- Model checker
 - Hoon-Collins CAD algorithm, a satisfiability algorithm (*Mathematica*)
- Learning algorithm
 - local polynomial interpolation

1.4 Why Numerical Code?

- M.G. Cox, P.M. Harris, E.G. Johnson, P.D. Kenward, and G.I. Parkin. *Testing the numerical correctness of software*. Technical Report CMSC 34/04, National Physical Laboratory, Teddington, January 2004.
- Showed numerical errors in NAG, IMSL, Microsoft Excel, Lab- VIEW and Matlab,
- Numerical specifications exist!
- Starting point for *cyber-physical systems*
- *Data-oriented* testing vs. *control* oriented
- The algorithms and models fit together extremely well!

Technical Approach

2.1 Models of Numerical Code

- Assume numerical code can be modeled as a function

$$f : \mathbb{R}^m \rightarrow \mathbb{R}^n$$

- (ignores non-termination)
- Decompose into n co-ordinate functions

$$f_i : \mathbb{R}^m \rightarrow \mathbb{R}, \quad i = 1, \dots, n$$

2.2 Medial Spheres Approximation

Decompose each $f_i : \mathbb{R}^m \rightarrow \mathbb{R}$ into **piecewise polynomial approximations** f_i^1, \dots, f_i^k over **m-dimensional spheres** S_1, \dots, S_k with centres c_1, \dots, c_k , and radii r_1, \dots, r_k

Looks like *Weierstrass Theorem* on polynomial approximation.

Piecewise methods tolerate **discontinuities**.

Correct approximation theory is *medial sphere approximation* (c.f. solid modeling)

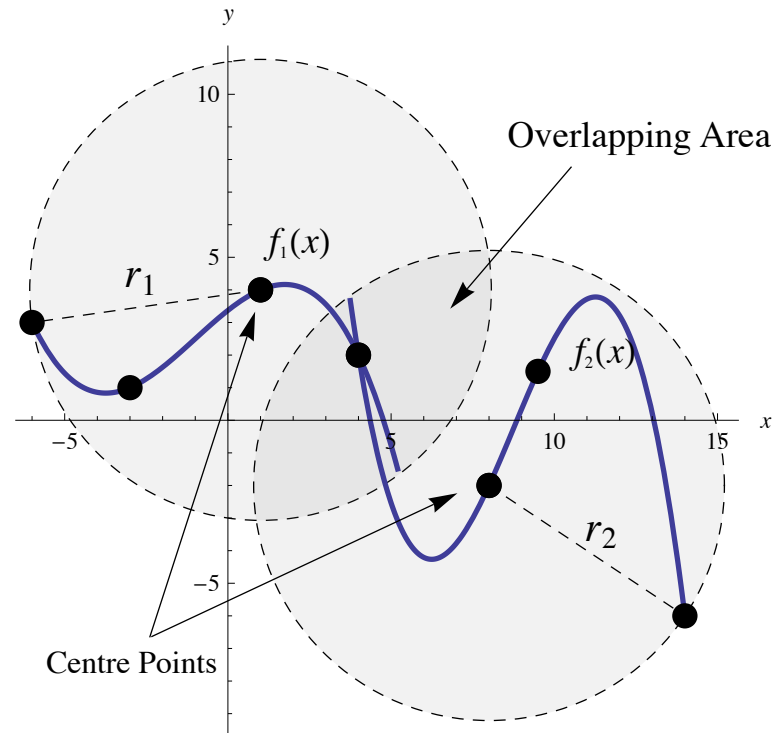
2.3 Approximation Support

- $(d+1)^m$ points in S needed to **uniquely determine** an m -dimensional degree d interpolating polynomial $p(S)$
- $x_1, f(x_1), \dots, x_{(d+1)^m}, f(x_{(d+1)^m})$
- $x_1, \dots, x_{(d+1)^m}$ are the **support** for **interpolant** $p(S)$
- Point can be **randomly placed** in S .
- $p(S) \neq 0$ tends to infinity outside S ,
 - so no extrapolation!
- Spheres S_i
 - overlap for smoothness,
 - disjoint for discontinuities

2.4 Non-gridded Support Sets

- A gridded approach to data sampling doesn't scale
- Exponential blowup in grid-point number with dimension size (m)
- Can't sample off-grid so might miss bugs

2.5 Non-gridded Piecewise Models



Two overlapping 2-dimensional local models (cubic approximations)

2.6 Model Refinement

- Every data point c becomes the centre of a sphere S_c
- Members of S_c are the $(d+1)^m - 1$ nearest members.
- As new points are added globally, spheres tend to shrink, improving their approximation accuracy.

2.7 Model Convergence

- Measure convergence of each sphere **locally** as an integral
- $\int_{S_{\text{new}}} p(S_{\text{old}}) - \int_{S_{\text{new}}} p(S_{\text{new}})$
- Compute this by **Monte Carlo approximation**
- Choose **least converged sphere as a breadth first search heuristic - minimise uncertainty**

2.8 Requirements Modeling

- Use Hoare triples $\text{pre}\{\text{code}\}\text{post}$
- pre and post are arbitrary first-order formulas (quantifiable!) over language $L(\mathbb{R})$ of real-closed fields.
- Tarski's Theorem “ $\text{FOL}(\mathbb{R})$ is decidable”
- Hoon-Collins CAD algorithm
Cylindric algebraic decomposition
- Doubly exponential time algorithm!
- Solvable for 6-8 free variables in practise.

2.9(a) What to solve?

- `pre` contains *invars* x_1, \dots, x_m
- `post` contains x_1, \dots, x_m and *outvars* x'_1, \dots, x'_n
- x' is post execution state of x , e.g.

$$x \geq 0.0 \{ \text{Newton-code} \} \mid x' * x' - x \mid \leq \varepsilon$$

Replace x'_i by $p_i(S)(x_1, \dots, x_m)$ in `post`, e.g.

$$x \geq 0.0 \{ \text{Newton-code} \} \mid p_1(S)(x) * p_1(S)(x) - x \mid \leq \varepsilon$$

for each sphere S for co-ordinate $1 \leq i \leq n$

2.9(b) What to Solve?

Solve for $x_1, \dots, x_m \in \mathbb{R}$ the formula

$\text{pre}(x_1, \dots, x_m) \ \&$
 $\langle x_1, \dots, x_m \rangle \in S^i, \text{ for } i=1, \dots, k \ \&$
 $\neg \text{post}(x_1, \dots, x_m, p_1(x_1, \dots, x_m), \dots, p_n(x_1, \dots, x_m))$

We can call CAD on this formula, and can ask for several solutions to x_1, \dots, x_m .

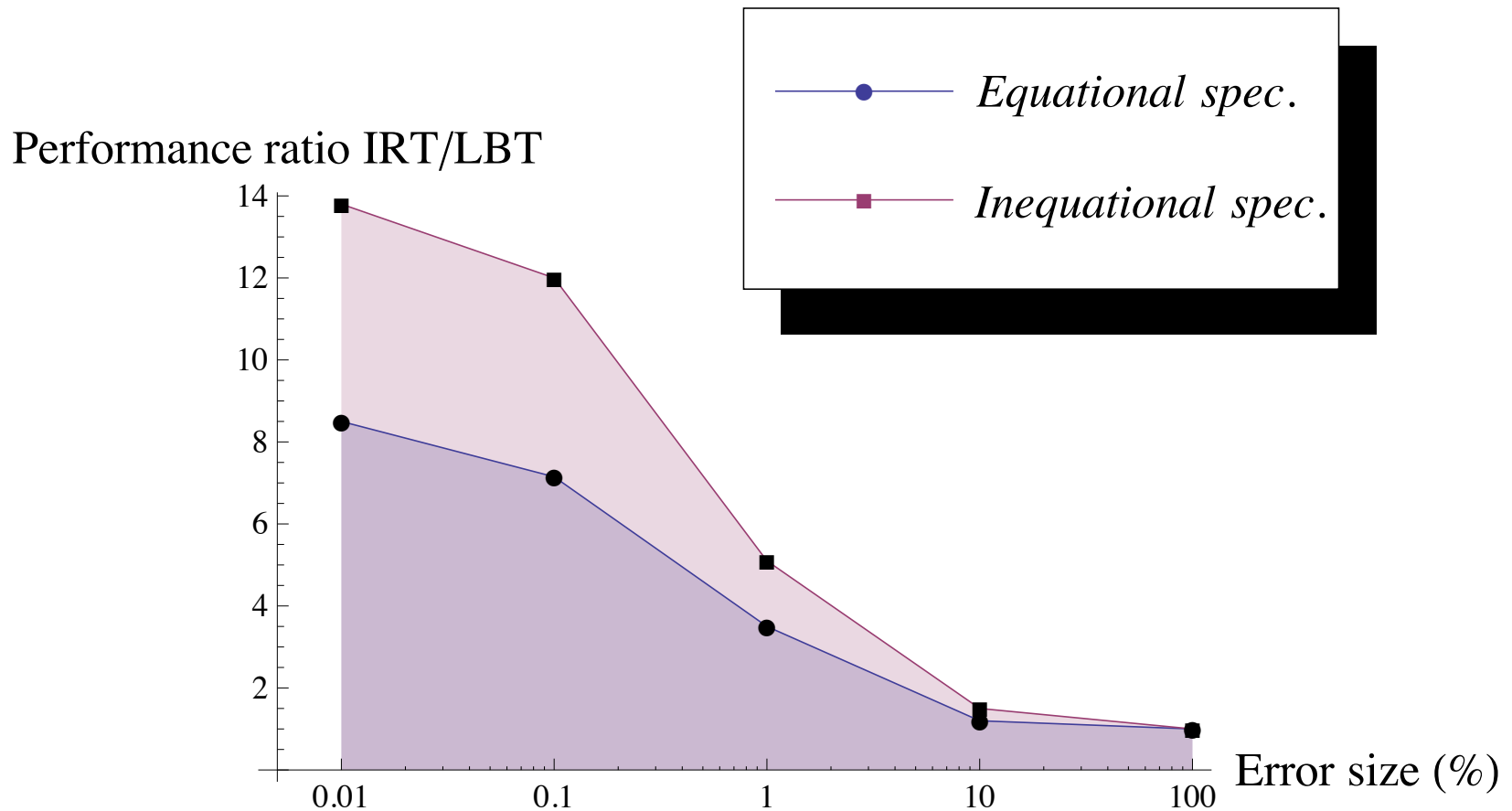
Use *k-wise testing* for large m , where $k < m$

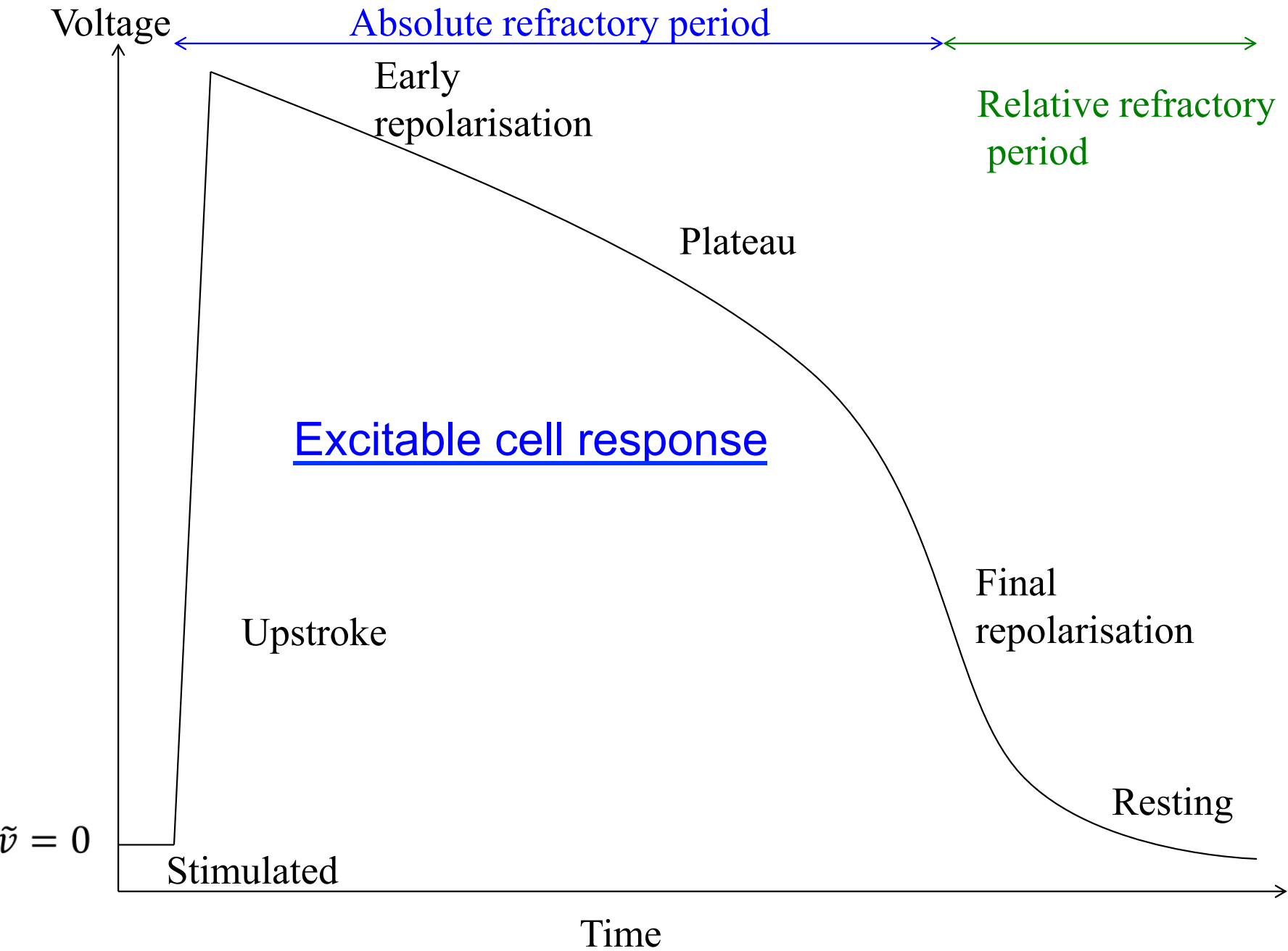
Part 3: Benchmarking Results

3.1 How to evaluate?

- Decided to benchmark against **random testing**.
- Small numerical algorithms are VERY fragile against mutations.
- Small mutation has large destructive effect.
- Built a **random use-case generator**
 - Randomly generated numerical functions
 - Associated formal specifications

3.3. Statistical Benchmarking against Random TCG

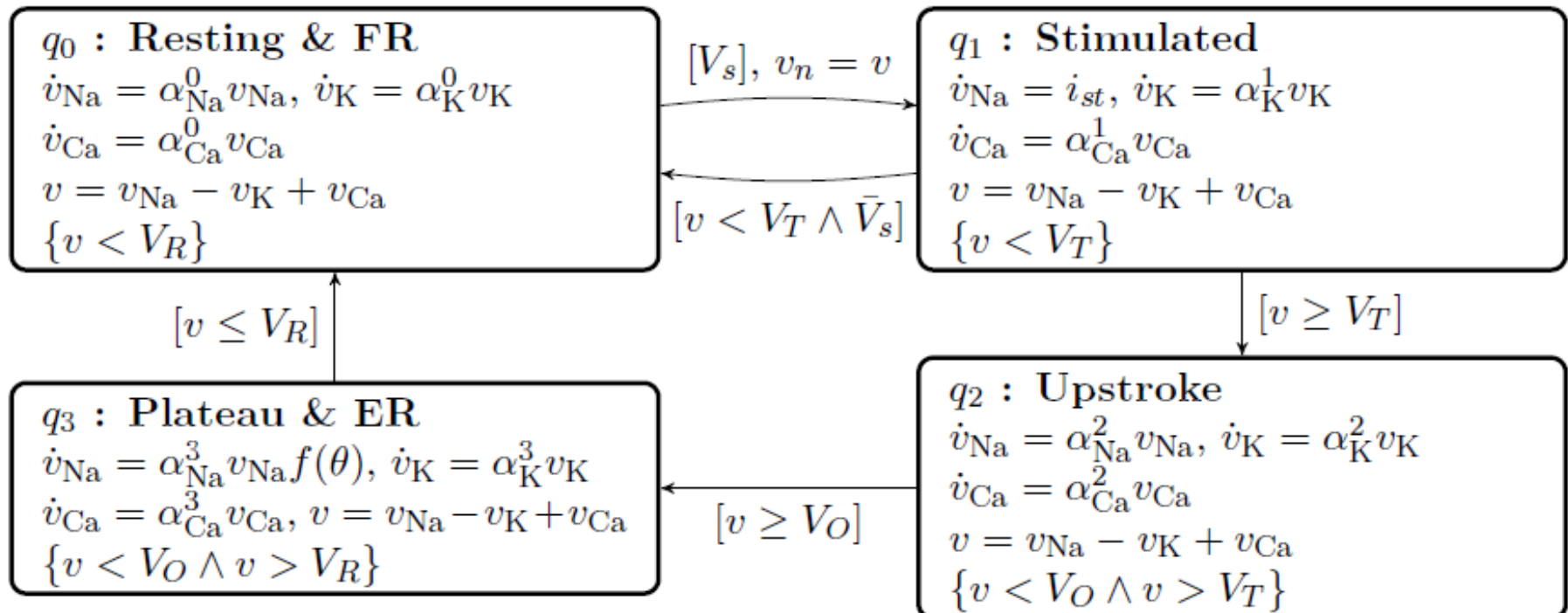




A Hybrid Automaton Model

The Luo-Rudy Guinea Pig Ventricular Cell model (LRd)

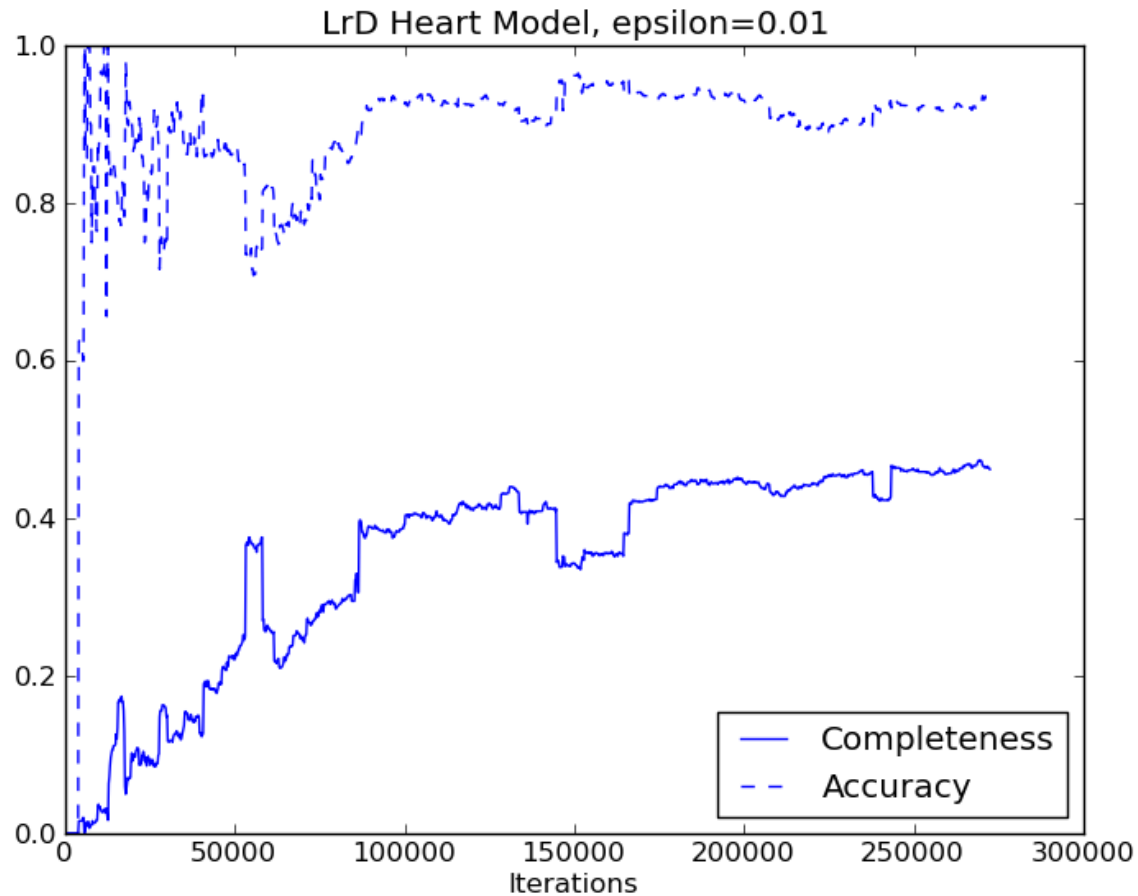
$$\theta = v_n/V_R \quad f(\theta) = 1 + 13\sqrt[6]{\theta}$$



Convergence Results (HyCGE learning algorithm)



KTH Datavetenskap
och kommunikation



4. Conclusions

- Computationally tractable case
- Good example of the LBT paradigm
- Interpolation “works” as inductive inference, especially due to continuity over \mathbb{R}
- Convincing benchmark results
- Provides insight into data-oriented LBT
- Thanks to the IsoLA Organisers!

Literature

Based on:

K. Meinke, *Automated Black-Box Testing of Functional Correctness using Function Approximation*, in: Proc ISSTA 2004

K. Meinke and F. Niu, *A Learning-Based Approach to Unit Testing of Numerical Software*, in Proc. ICTSS 2010

K. Meinke and F. Niu: *An Incremental Learning Algorithm for Hybrid Automata*, KTH-CSC technical report, 2013.

D. Lundberg, R. Ansin, *Automated Inference of Excitable Cell Models as Hybrid Automata*, Bachelor's Thesis, SA104X, KTH Royal Institute of Technology, 2013.