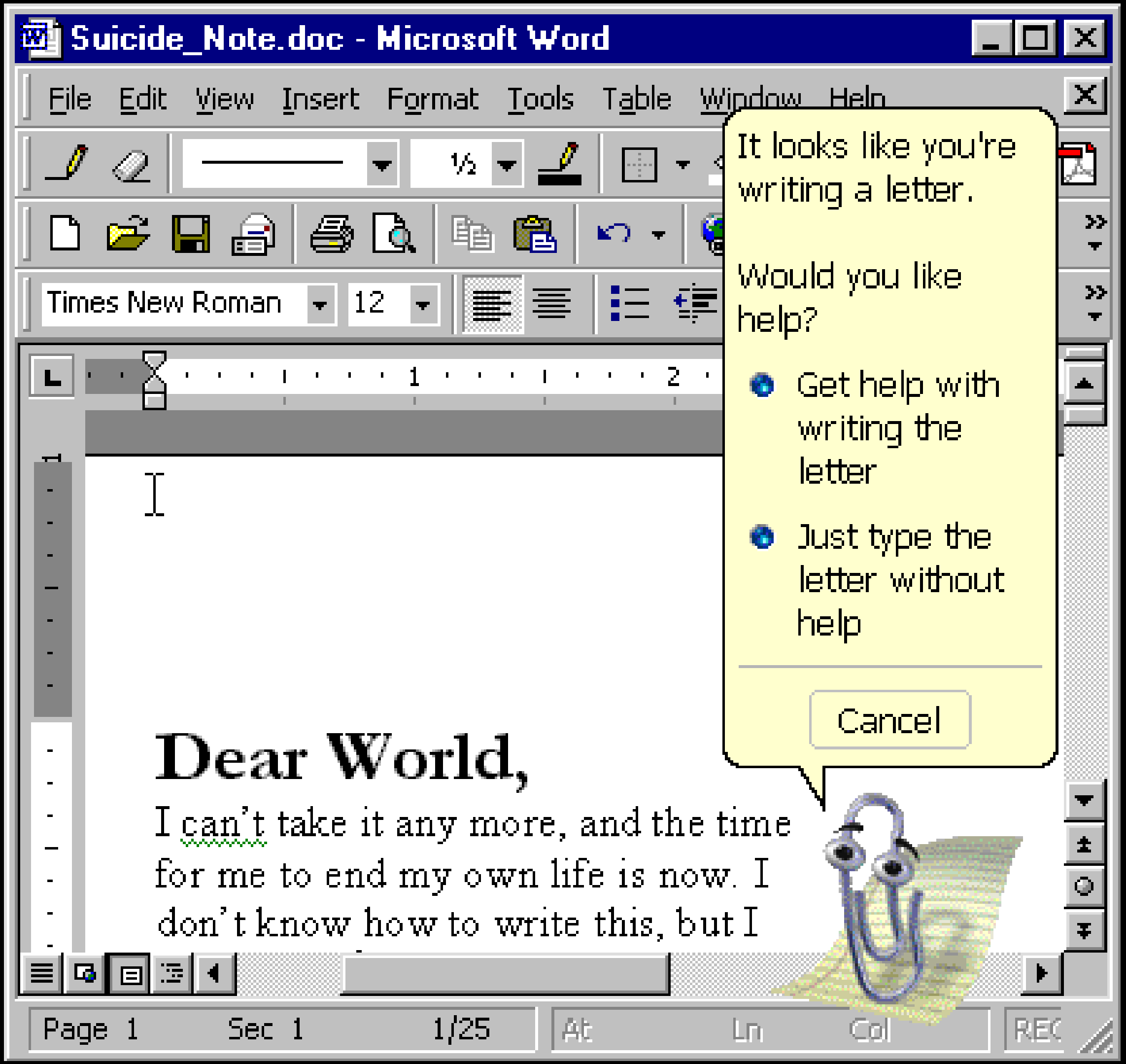
Generative Tools for Hybrid Systems

Jonathan Sprinkle, Ph.D.
University of California, Berkeley

Overview

- Introduction
- Motivation
- Backgrounds
  - Domain-Specific Modeling
  - Hybrid Systems
- What has been done
- Looking forward
- Conclusions

"Help"


Suicide_Note.doc - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Times New Roman
12
It looks like you're writing a letter.
Would you like help?
Get help with writing the letter
Just type the letter without help
Cancel
Dear World,
I can't take it any more, and the time for me to end my own life is now. I don't know how to write this, but I
Page 1 Sec 1 1/25 At Ln Col REC

Writer's Block

- What is more daunting than a blank page, and an unfamiliar task (language, topic, program, etc.)
- How can you assimilate bits of pieces of information (stored in your head, distributed throughout your design/idea, and informally stated at best) into a coherent concept understandable to your end audience?

Mythbusters!

The myth that some people come away believing, when exposed to the notion of a formal language, is that a "formal language" is a formal-looking language; that any language that contains lots of Greek letters and mathematical symbols is formal.

— *David Harel*, *Bernard Rumpe*, "Syntax, Semantics, and all that Stuff"

Why Model Domains?

- Domain modeling can be
  - Formal
  - Intuitive
  - *Useful*

Do you know what the funny thing is about domain modeling? It's the little differences.

Example(s)

- PowerPoint
  - Domain: Visual Presentation
- Excel
  - Domain: Accounting/number crunching
- MATLAB
  - Domain: Discrete systems
- $\LaTeX$
  - Domain: Typesetting (sub: Academic papers, books, posters…)

- Problems:
  - *How long does it take to create one of these environments???*
  - *What happens if you try to use one of these environments for something it was not intended???*
  - *What about creating domains for* non-traditional *systems???*

Creating Domain-Specific Modeling Environments (DSMEs)

Berkeley
University of California

- A *working* application for system design
- A *customized* modeling environment which is a *restricted* input layer that enforces some meaning
- An implementation *reflecting* a domain's familiar and consistent
  - methodologies
  - notation
  - semantics

- An *efficient* user interface

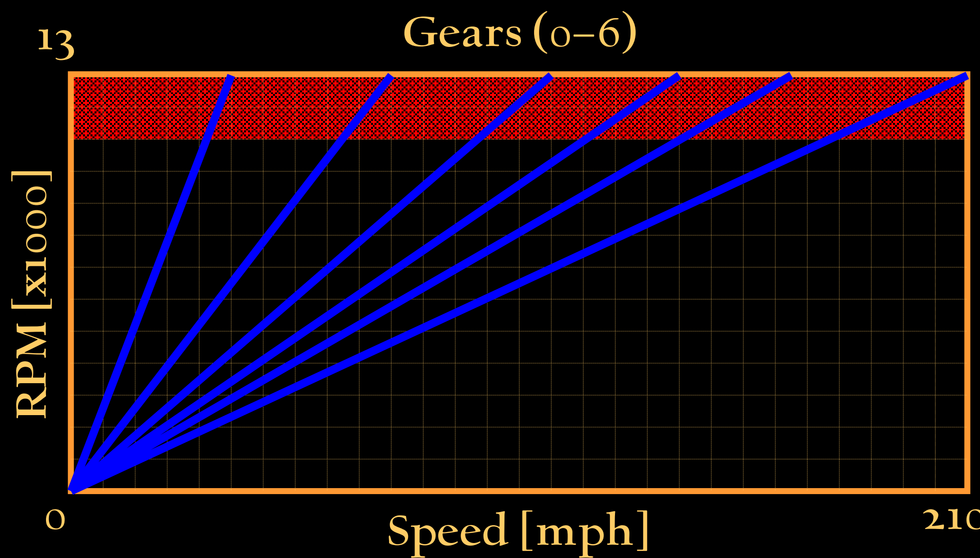
Hybrid Systems

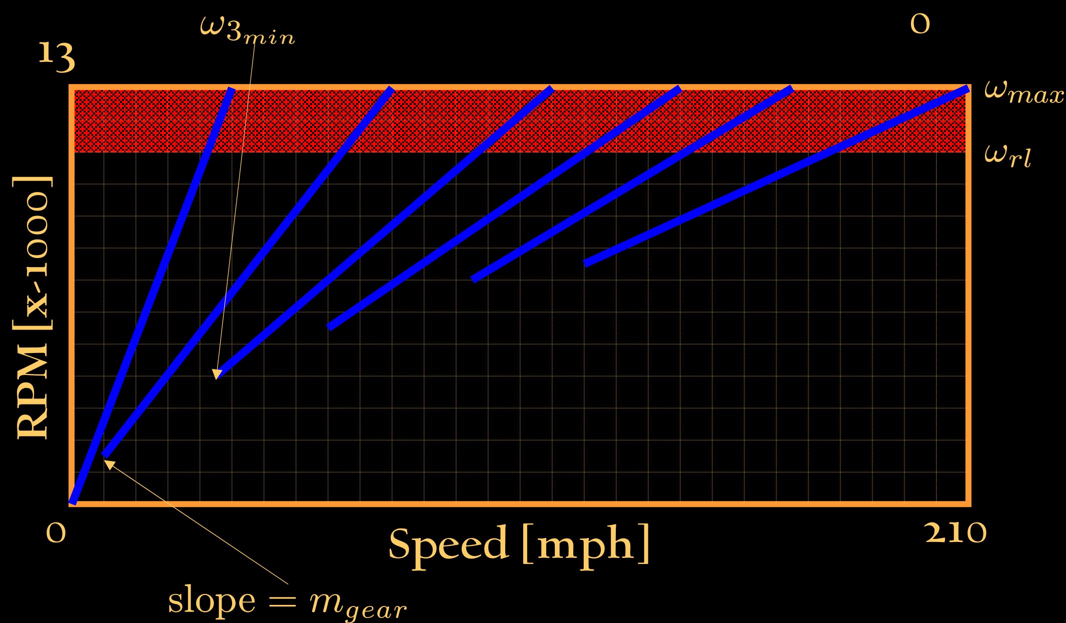
- An emerging, complex, engineering discipline
- Systems that are described both by
  - Discrete states of operation (e.g., modes)
  - Continuous dynamics within each discrete state

Example Hybrid System

Automobile velocity

- Shifting gears allows higher speeds before damaging engine (a.k.a. "redlining")
- However, not all gears function well at low RPM, requiring a certain speed before their use

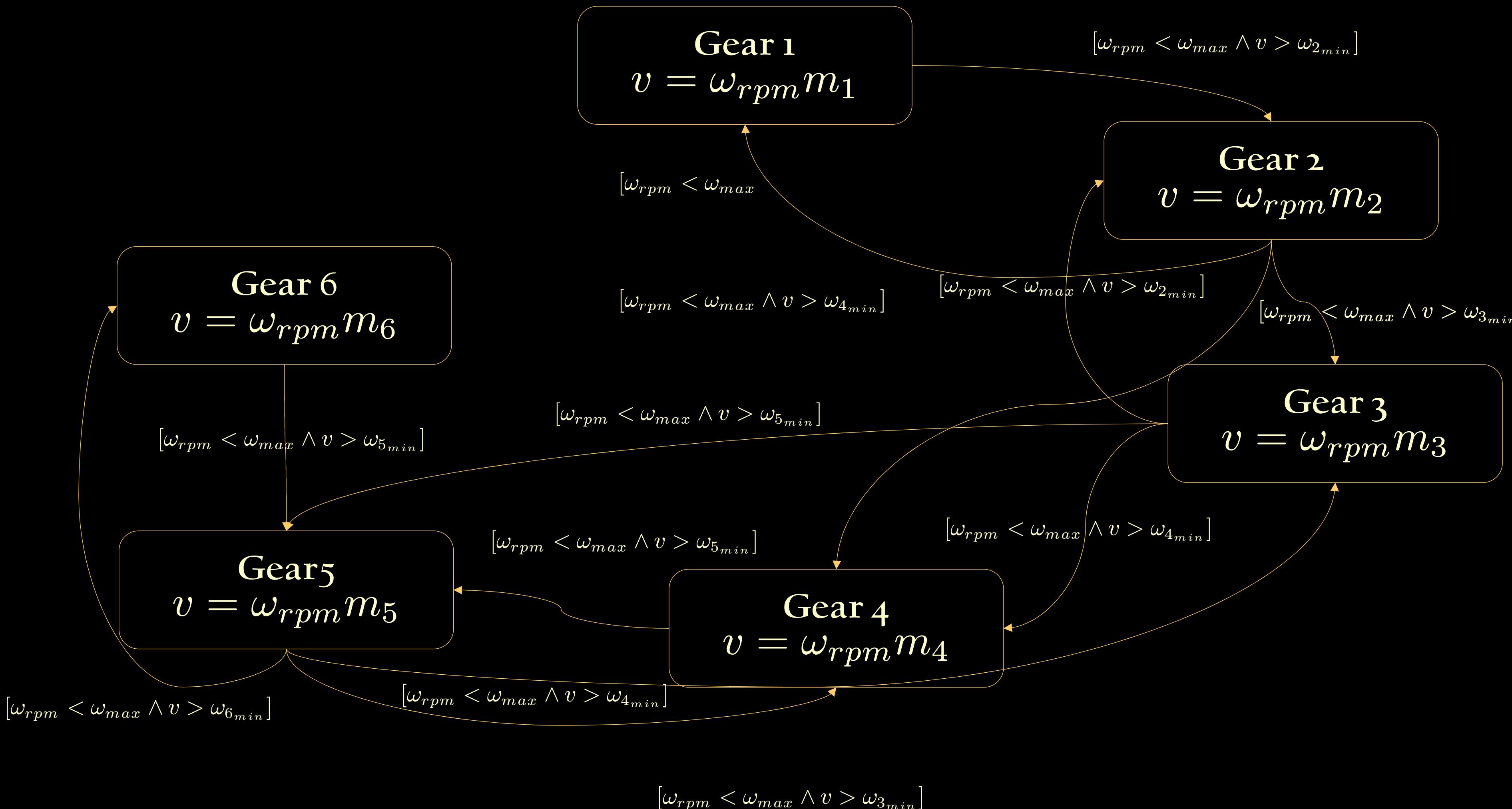




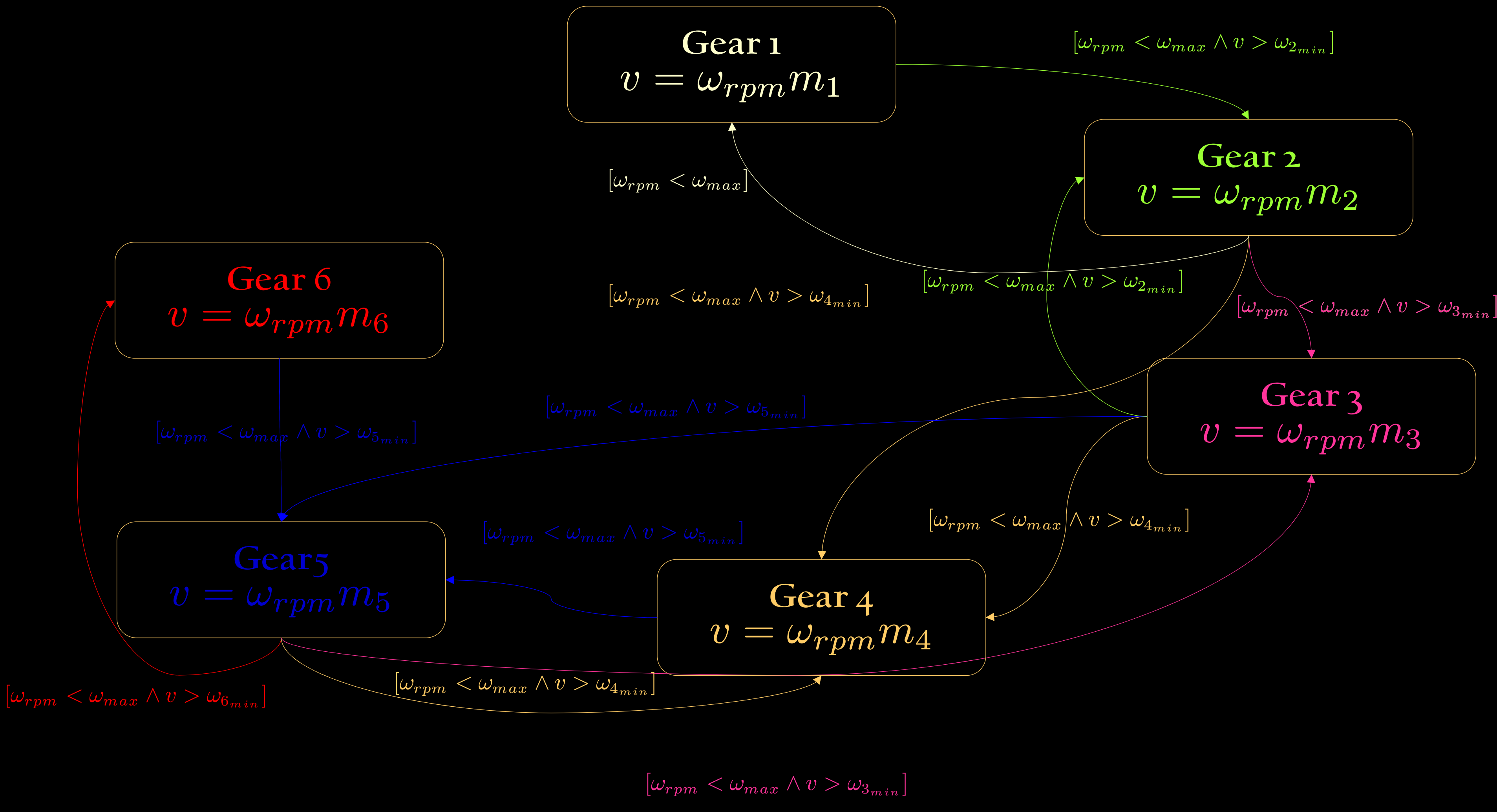
Constrained gearbox

- "Safe" zones for each gear
- Limited shifting, due to safe zones
- Requires a smart controller for automatic transmissions

Mathematical Specification of Transmission System



Mathematical Specification of Transmission System

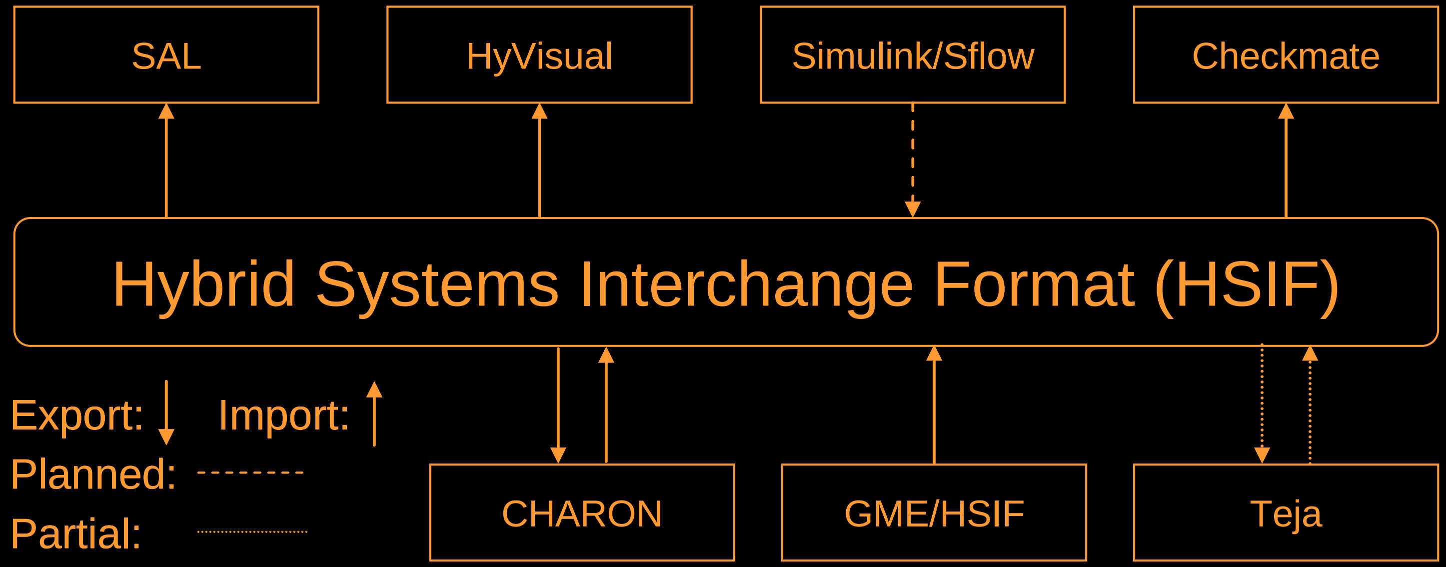


Hybrid Systems Tools

- Modeling
  - Describe the system, it's constraints, some portions of the controller
- Controller synthesis
  - Generate switching criteria, guards, etc., based on constraints
- Verification/Validation
  - Assert or contradict that the controller satisfies the constraints
- Code generation
  - Actually implement the controller in an embedded system
- No one tool can do all of this?
  - So, what about interchanging models between tools?

HSIF

- The Hybrid Systems Interchange Format (HSIF) was designed to satisfy the first portion (system spec.)



HSIF Modeling Language

- As a graduate student, I created the HSIF Modeling Environment (HSIF-ME)
  - a domain-specific graphical modeling tool for the hybrid systems community
  - specification very similar to mathematical definition (as proposed by Lygeros, Simic, et al.)
  - generated several formats, for the tools that provided their syntax and semantics
  - easier to use than the specialized verification/validation simulation tools (for the most part)
- A very lightweight tool (can exist without any other components) for system description

HSIF Problems

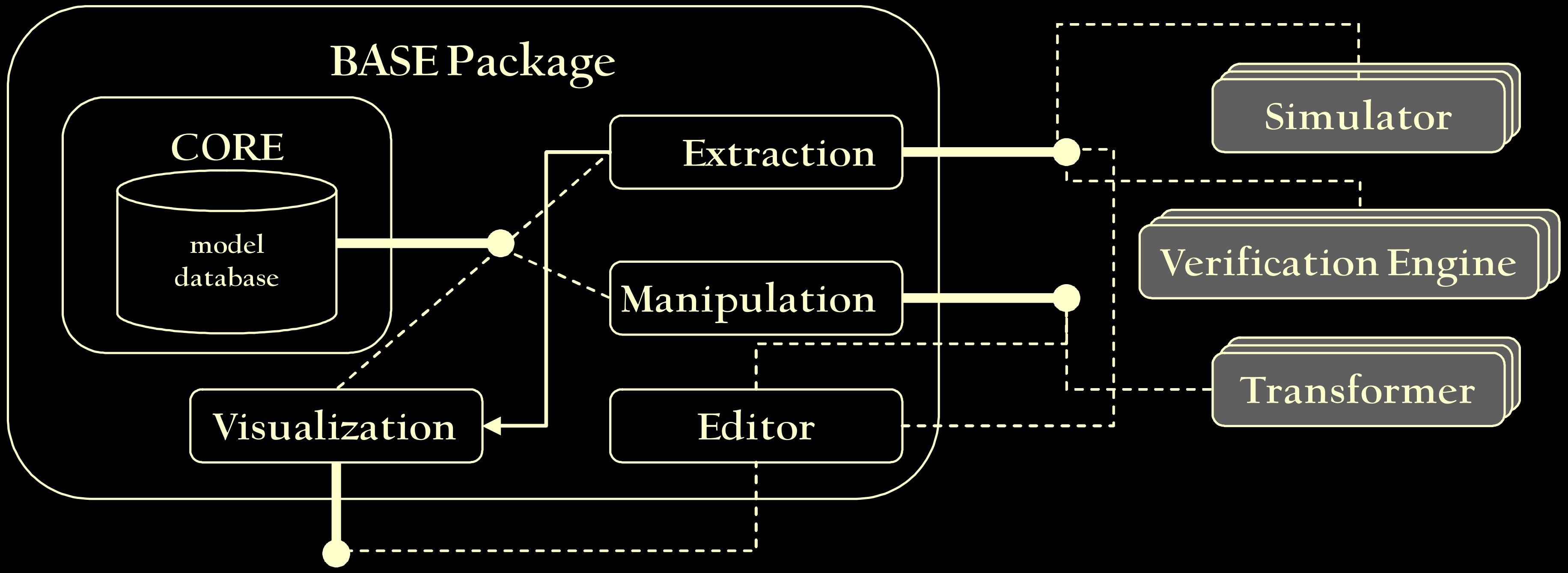
- Design by committee
  - Too many tool-specific syntax entries
  - Unclear semantics for some syntaxes
- Many people wanted to be involved, few wanted to put up the work to match their rhetoric ☺
  - Resulted in me doing all the work
  - This is why so many routes are "planned"

Looking forward…

- How to address these problems:
  - Design by committee, unintuitive syntaxes, semantic interchange issues
- How to maintain these goals:
  - Intuitive modeling interface, tool interoperability
- How to improve basic tasks
  - Utilize state-of-the-art simulators, provide error bounds on event detection
- How to take advantage of emerging applications
  - BioSPICE, pursuit/evasion games, reachability calculations, space vehicle control (NASA H&RT)

Research Proposal

- A self contained facility, which can interchange components with more sophisticated tools



Scientific details

- How can we specify the semantics of the component interfaces?
  - Can the approach of IDL be taken, but abstract equation solving techniques rather than language/OS impl?
- What does it mean to deploy a totally abstract system?
  - Can we ship a version that will interchange with Matlab, as well as Mathematica, as well as a standalone C++ app, and dependably interact with the same models?
- How should we manage semantic interoperability?
  - Can we accept some mismatch in execution styles, and if so, how much mismatch results in incorrect roundtripping or incorrect execution strings?

Technical Details

- What language should we choose for implementation?
  - Python, Java, run on many platforms, not so fast (although with JNI maybe faster)
- Can we accept certain platform requirements for certain components (e.g., verifiers may work only in Linux for some components)

Why is tool interoperation hard?

- Hybrid systems tools share a common ontology
  - Hybrid Automata
  - Events and Transitions
  - Equations
- Common semantics with similar ontology
  - Flow vs. Differential equations
  - Discrete States vs. Locations
- Discrepant semantics with similar/common ontology
  - Global and local variable precedence
  - Model of computation discrepancies

Tool A || Tool B

A == A

sem(B) == sem(ß)

sem(C) != sem(C)

What steps are underway?

- Weekly discussion with leading experts at Berkeley
  - Shankar Sastry, Edward Lee, Tom Henzinger, and their students
- Interactions with previous participants
  - Vanderbilt, Penn, agree with need for new revisions
- Collaboration with industry to determine goals/constraints
  - Ford, GM, both require Matlab/Simulink interoperability for existing models

Conclusions

Berkeley
University of California

- Current state of the art is lightweight tool-abstracted interface format
- Desired research tool is similarly lightweight, but abstracted more by semantic requirements than desired working tools
- Tools still drive the nature of execution and development, but the research topics (especially biological ones) promise to provide required funding for the tool development

Questions?

“Well HAL, I’m damned if I can find anything wrong with it.”
“Yes. It’s puzzling. I don’t think I’ve ever seen anything quite like this before.”
-- *2001: A Space Odyssey*