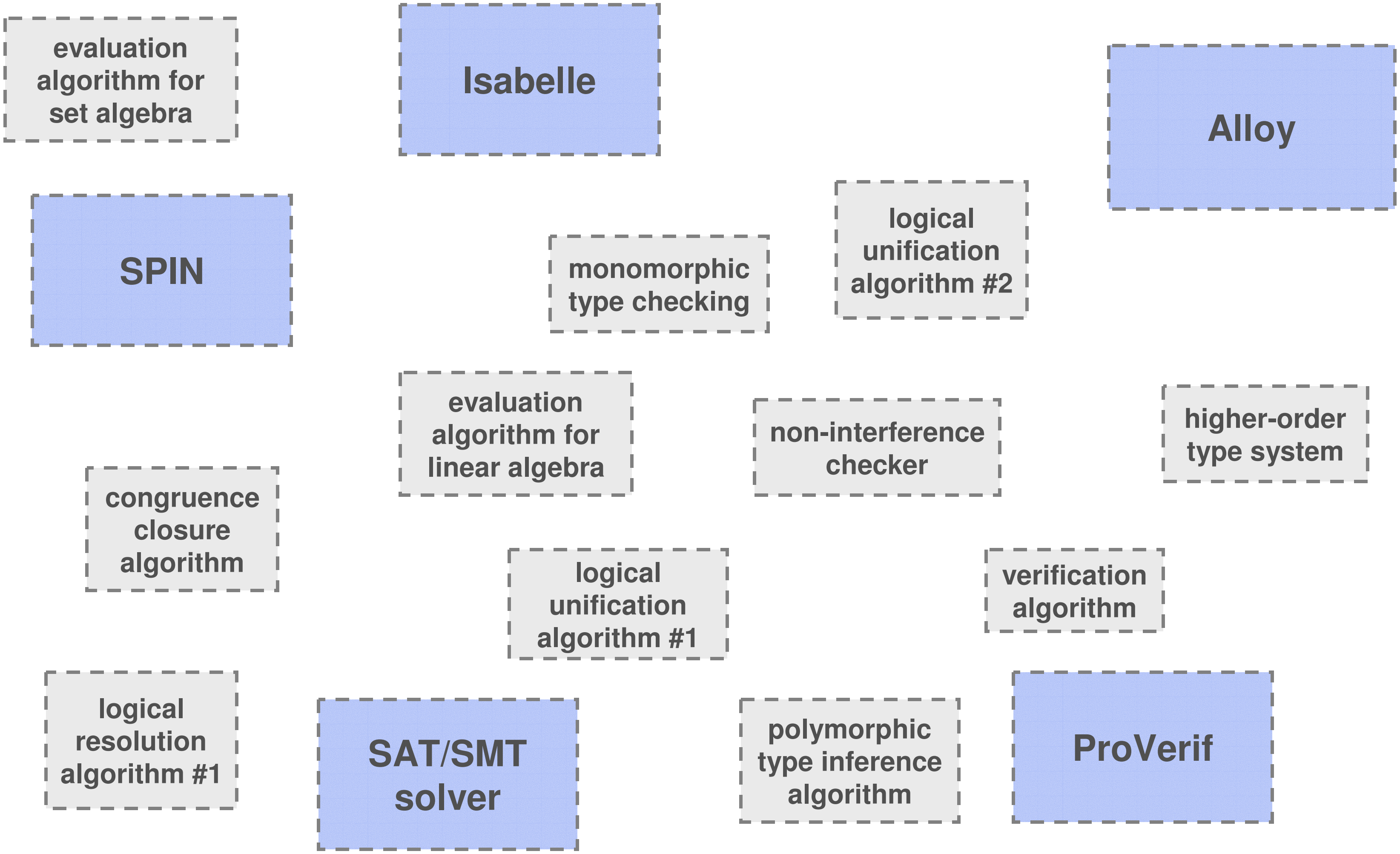
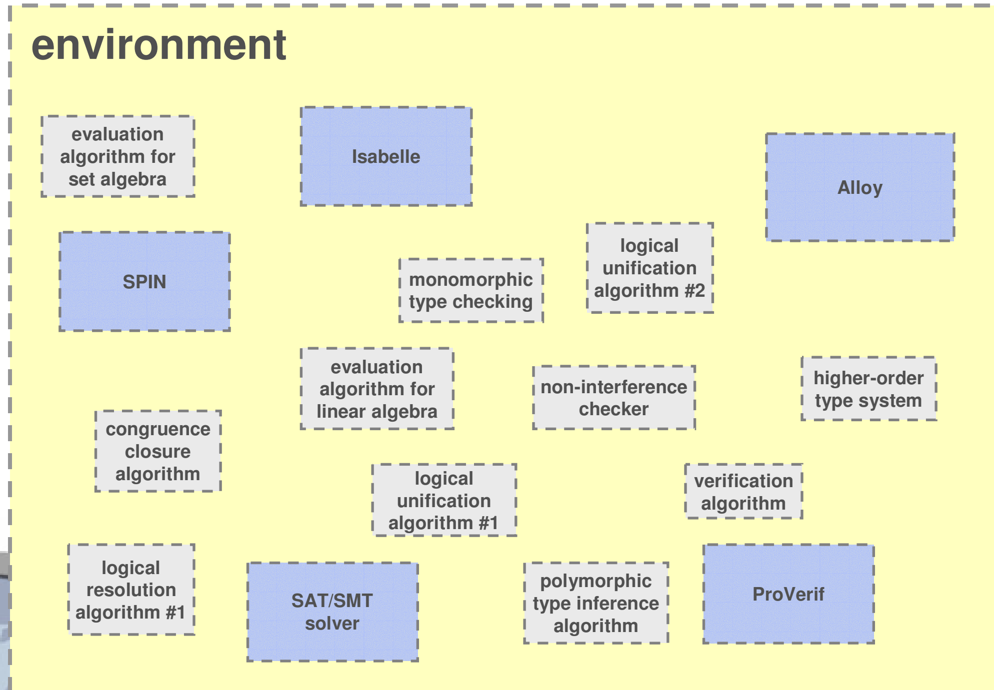
Accessible Integrated Formal Reasoning Environments in Classroom Instruction of Mathematics

Andrei Lapets
Boston University

May 8, 2012

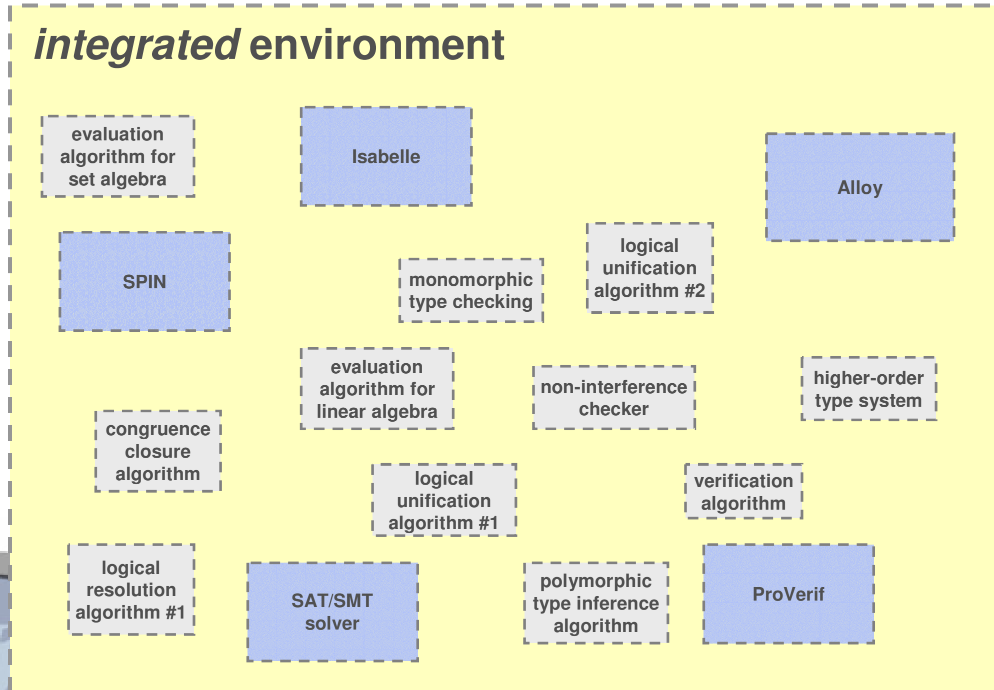


many tools and techniques have been developed by the programming languages, formal verification, and model checking communities



end-user
- engages in formal reasoning tasks

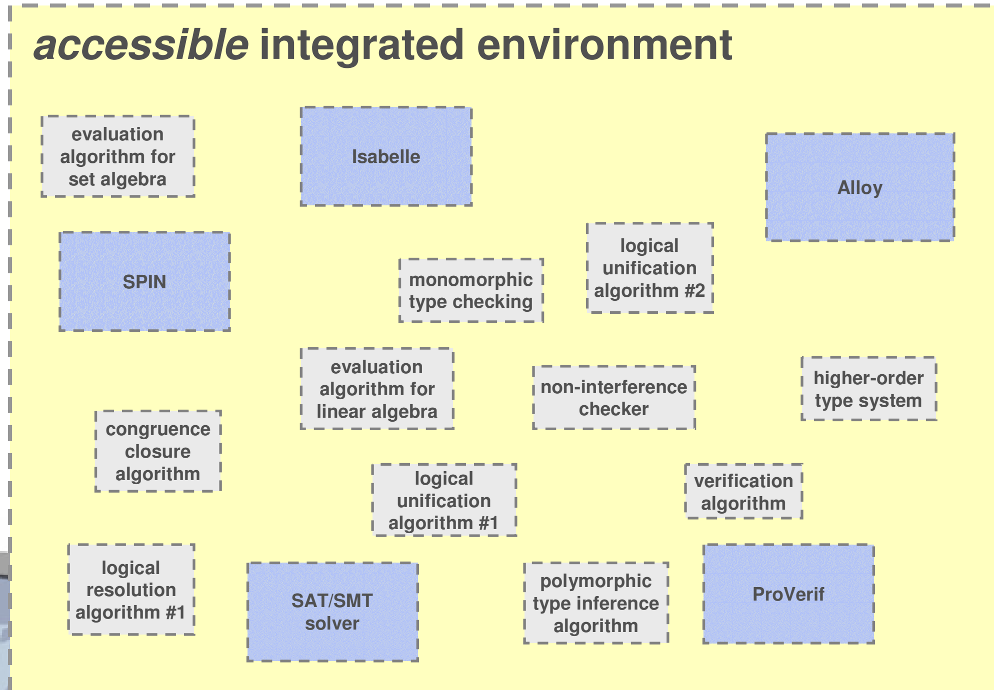
- a single environment can…
  - allow users to leverage **multiple** tools and techniques using a uniform, familiar interface and representation
  - make setup easy, or (if web-based) unnecessary
- these benefits may lead to more widespread utilization of existing tools and techniques



end-user
- engages in formal reasoning tasks

– focus is ***not*** on rigorous formal safety or correctness of integration of components
  - most formal models of domains are *incomplete*
  - however, two approaches are complementary

– value of integration: an automated **interactive** environment with multiple kinds of **instant feedback** identifying problems for users



end-user
- engages in formal reasoning tasks

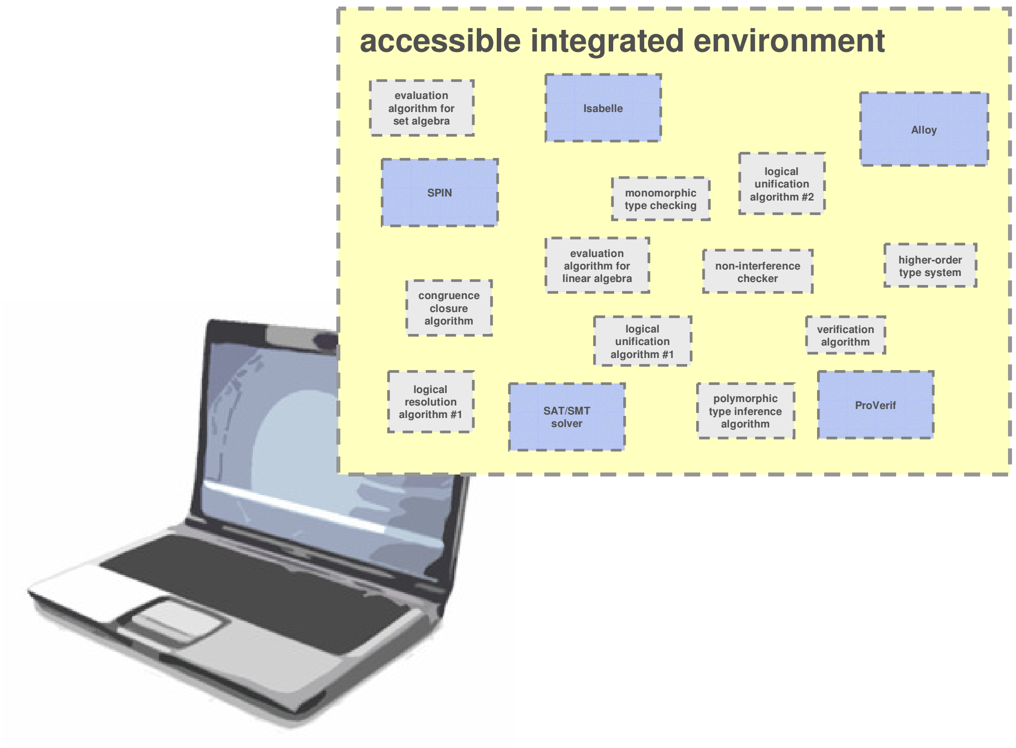
– no setup; no special environment needed
– familiar or conventional domain-specific syntax
– interactive, immediate feedback (guidance, results, validity)
– at least some feedback for partial arguments
– flexibility with regard to level of detail

in this work we are developing:

- a proposed collection of **conventions** and **practical tools** for building, instantiating, and delivering to end-users accessible and integrated formal reasoning environments

- a **context** for **posing questions** about the integration of automated formal algorithms and tools with one another and with other supporting components *we assume that there exist three user roles (possibly overlapping) that an infrastructure for accessible integrated environments must accommodate*

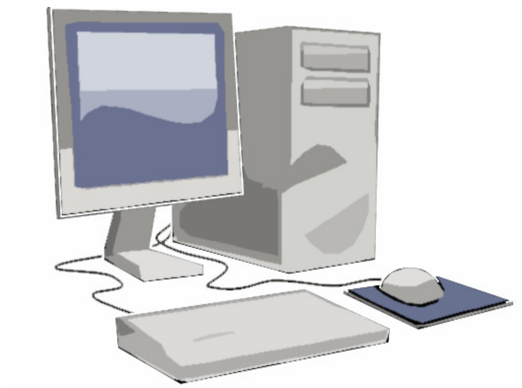
formal systems expert
- implements integrated algorithms
- implements translations to other systems



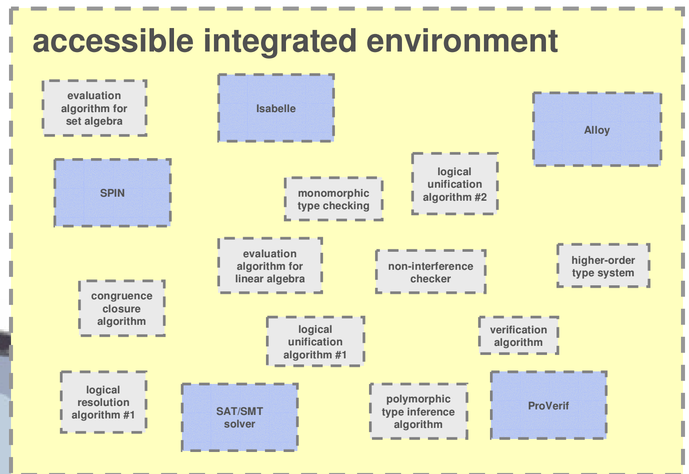
administrator/domain expert
- instantiates domain-specific libraries
- authors/ curates library contents
- manages environment/embeddings

end-user
- engages in formal reasoning tasks

these roles may correspond to more specific user types in particular application domains, such as classroom instruction



formal systems expert
- implements integrated algorithms
- implements translations to other systems

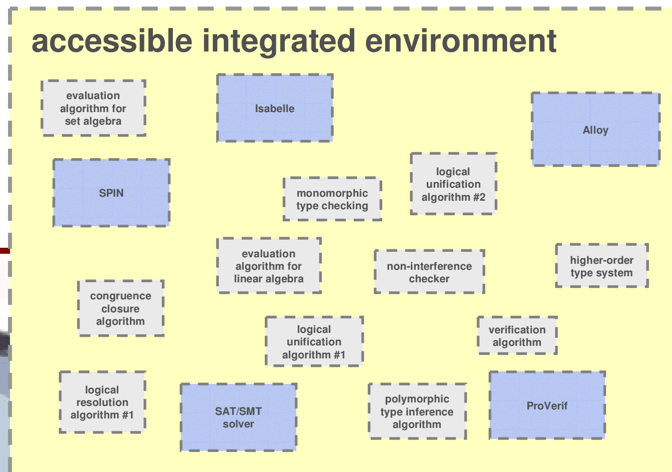


enrolled student
- uses environment to complete assignments

course instructor
- authors lecture notes w/ examples
- assembles assignments
- specifies available propositions

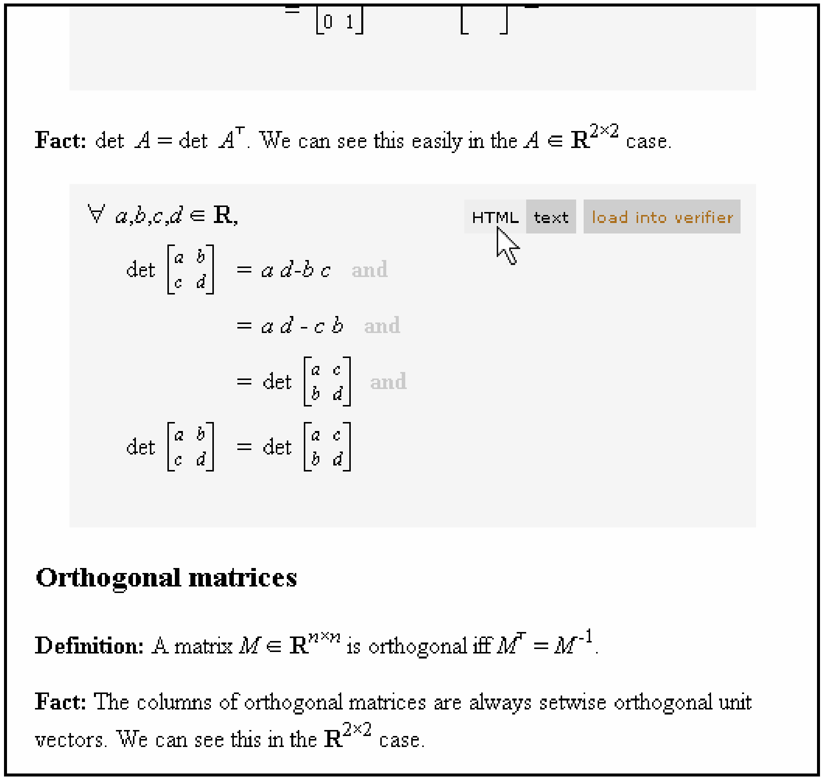
we briefly describe how an end-user might experience such an environment by looking at a prototype used in an undergraduate mathematics course

formal systems expert
- implements integrated algorithms
- implements translations to other systems



enrolled student
- uses environment to complete assignments

course instructor
- authors lecture notes w/ examples
- assembles assignments
- specifies available propositions


Fact: $\det A = \det A^{\top}$. We can see this easily in the $A \in \mathbf{R}^{2\times 2}$ case.

HTML text load into verifier

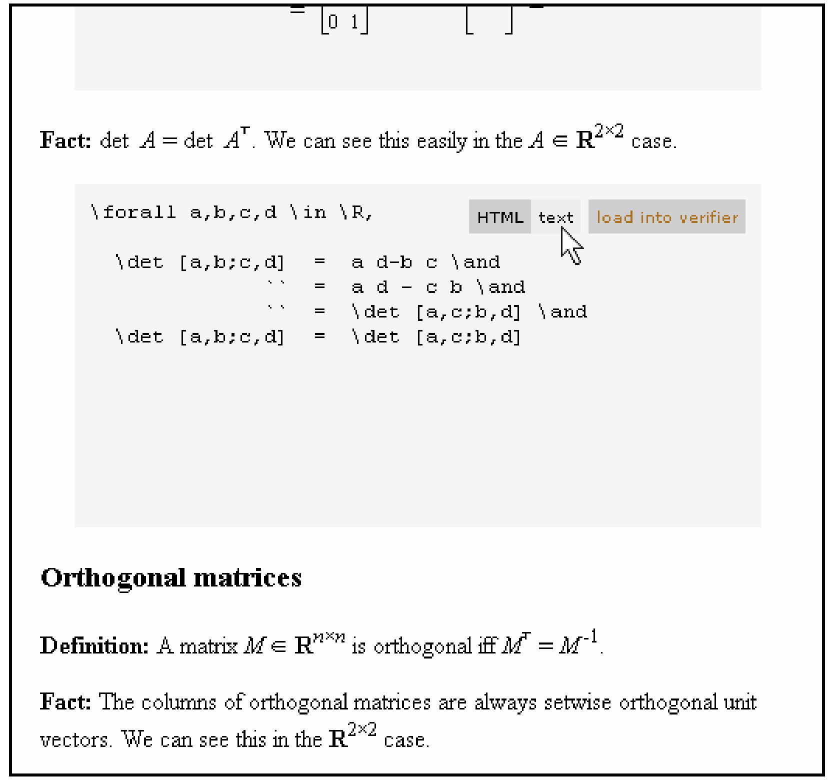
$\forall\ a,b,c,d \in \mathbf{R}$,

$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = a\,d - b\,c \quad \text{and}$$
$$= a\,d - c\,b \quad \text{and}$$
$$= \det \begin{bmatrix} a & c \\ b & d \end{bmatrix} \quad \text{and}$$
$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \det \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

Orthogonal matrices

Definition: A matrix $M \in \mathbf{R}^{n\times n}$ is orthogonal iff $M^{\top} = M^{-1}$.

Fact: The columns of orthogonal matrices are always setwise orthogonal unit vectors. We can see this in the $\mathbf{R}^{2\times 2}$ case.


Fact: $\det A = \det A^{\top}$. We can see this easily in the $A \in \mathbf{R}^{2\times 2}$ case.

HTML text load into verifier

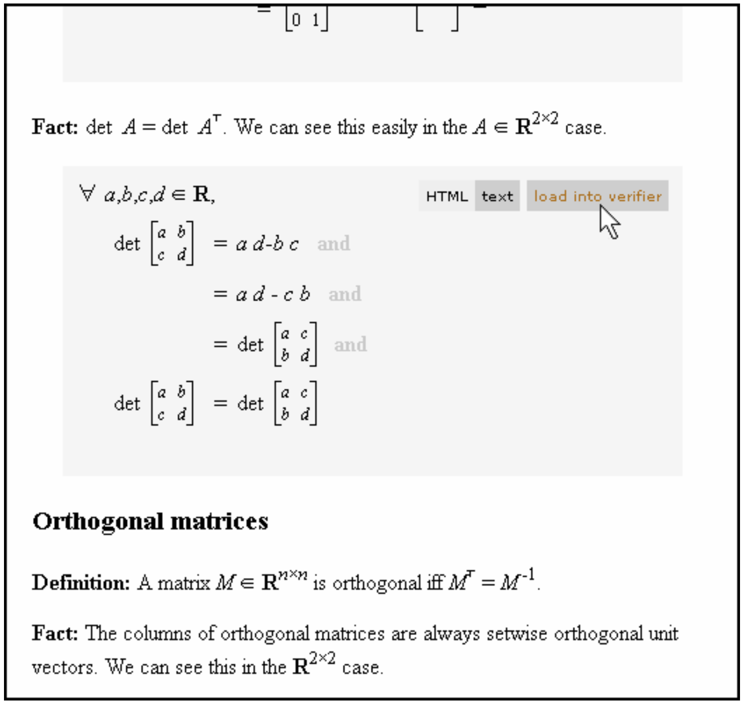
```
\forall a,b,c,d \in \R,

  \det [a,b;c,d]   =   a d-b c \and
              ``   =   a d - c b \and
              ``   =   \det [a,c;b,d] \and
  \det [a,b;c,d]   =   \det [a,c;b,d]
```

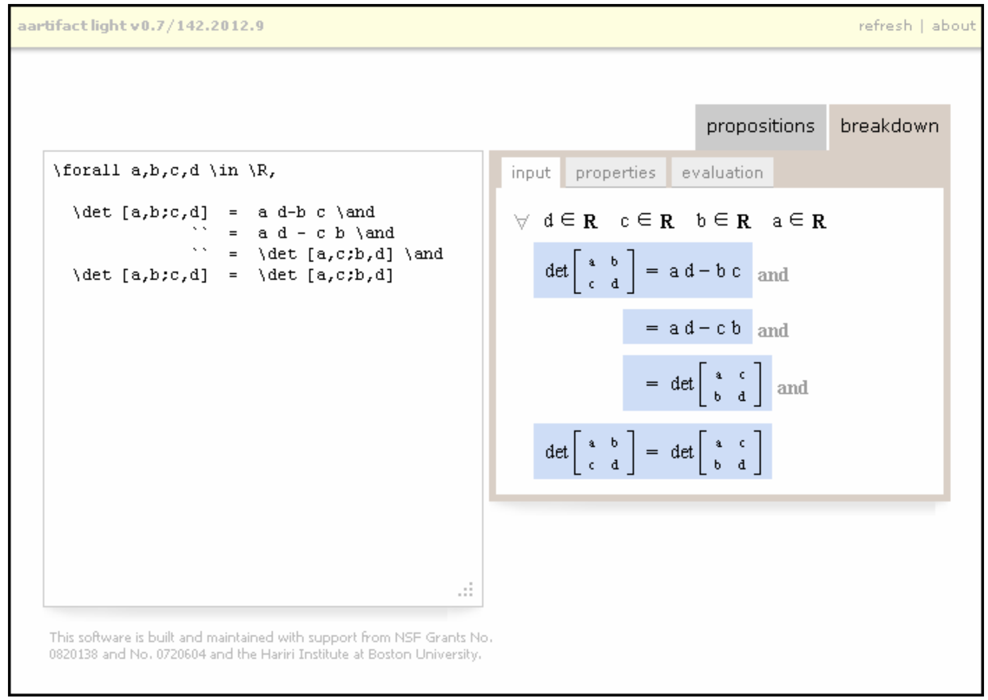
Orthogonal matrices

Definition: A matrix $M \in \mathbf{R}^{n\times n}$ is orthogonal iff $M^{\top} = M^{-1}$.

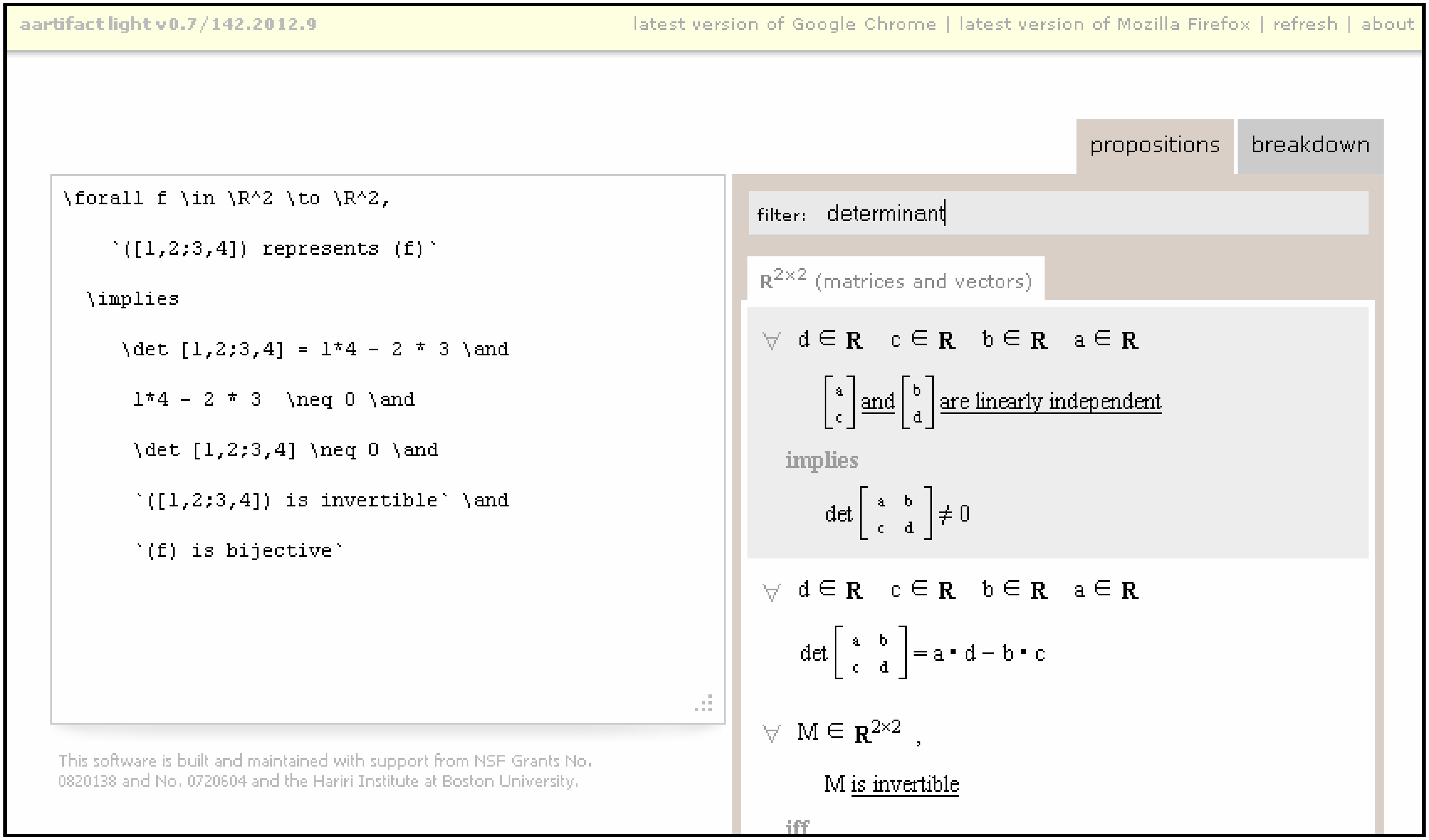
Fact: The columns of orthogonal matrices are always setwise orthogonal unit vectors. We can see this in the $\mathbf{R}^{2\times 2}$ case.

online course notes contain verifiable arguments that can be viewed by students either in a friendly format or as verifiable formal syntax

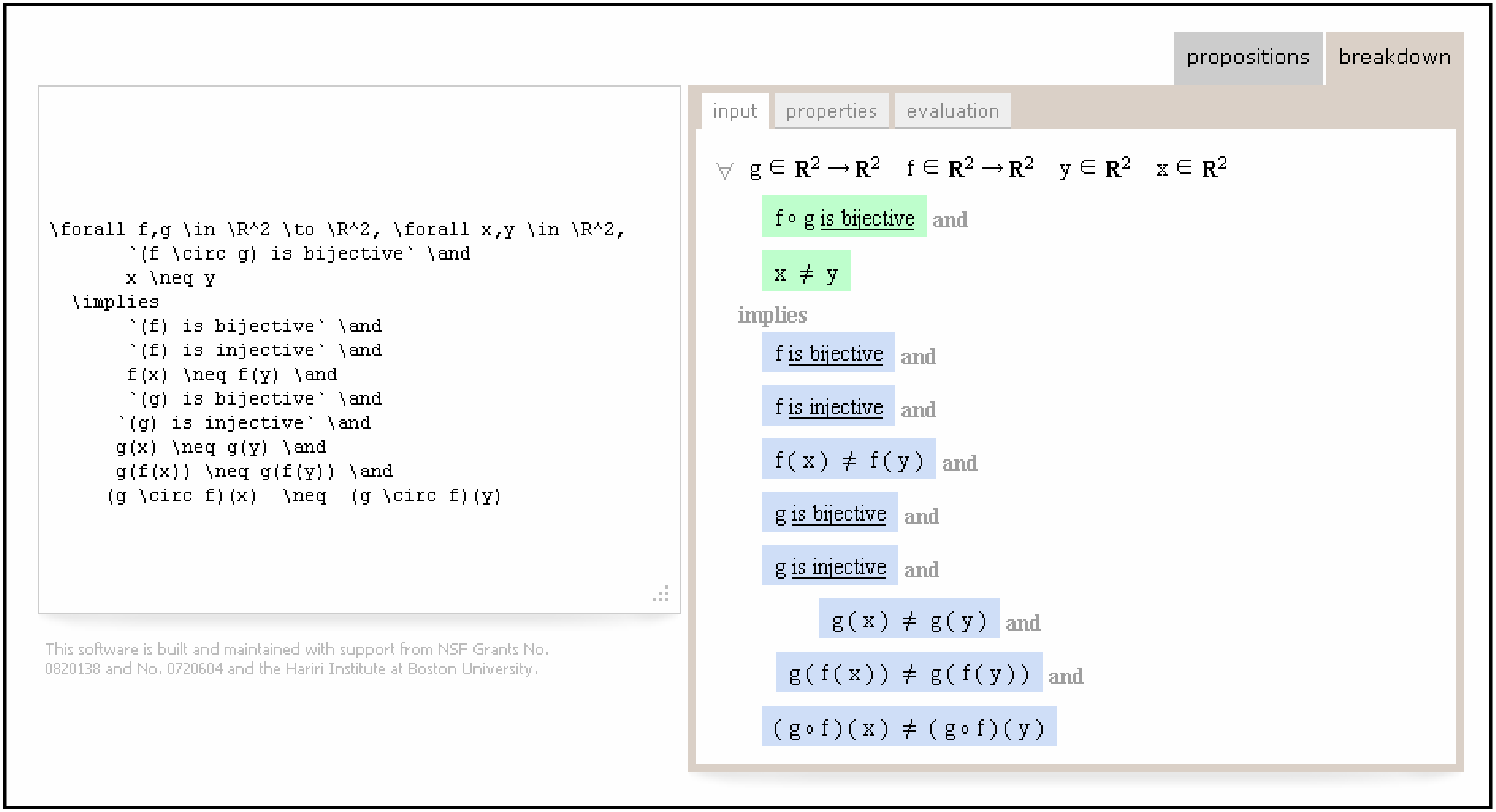




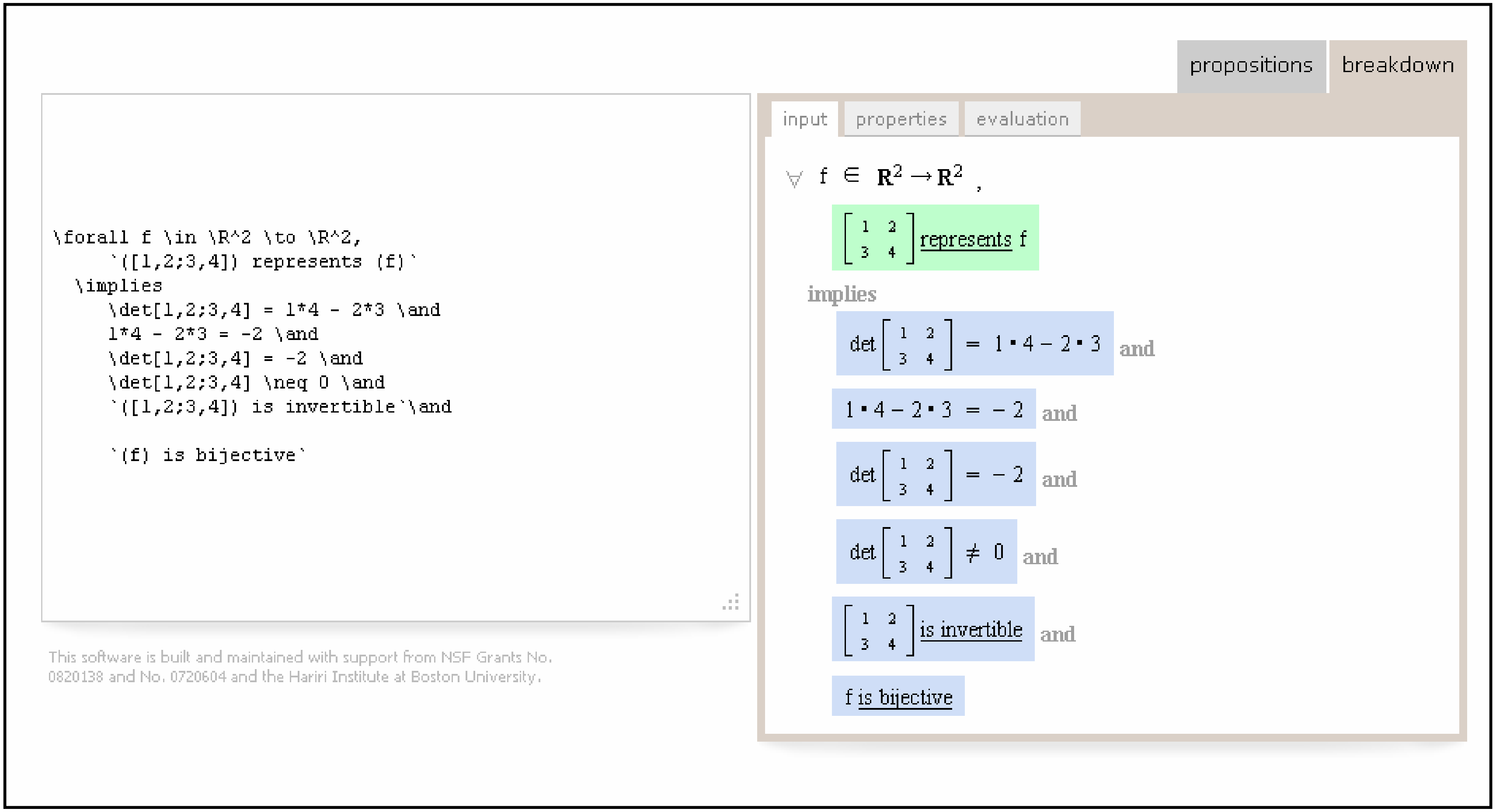
verifiable formal arguments included in the course notes can be loaded instantly into the integrated environment



students can view and explore the propositions made available for an assignment by the instructor

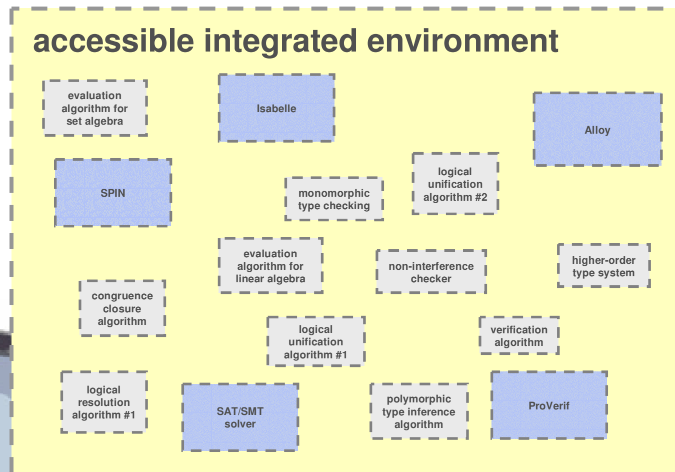


actual examples of verifiable formal arguments assembled by students

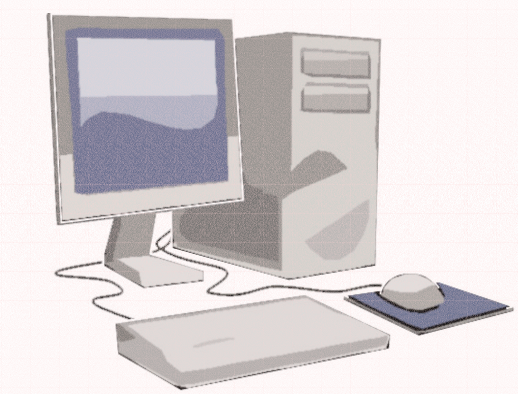


actual examples of verifiable formal arguments assembled by students *the supporting tools for a course instructor are currently a work in progress*

formal systems expert
- implements integrated algorithms
- implements translations to other systems

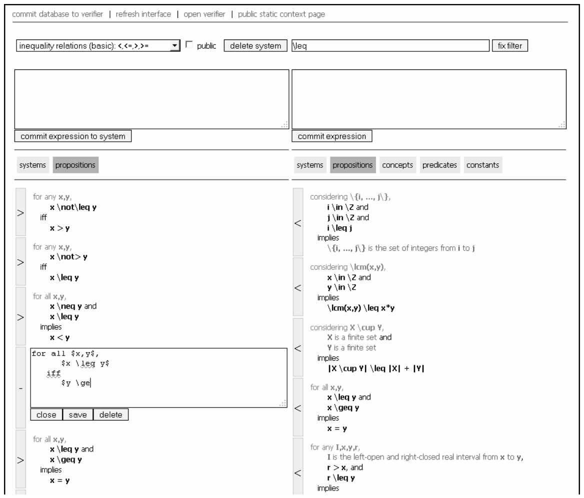


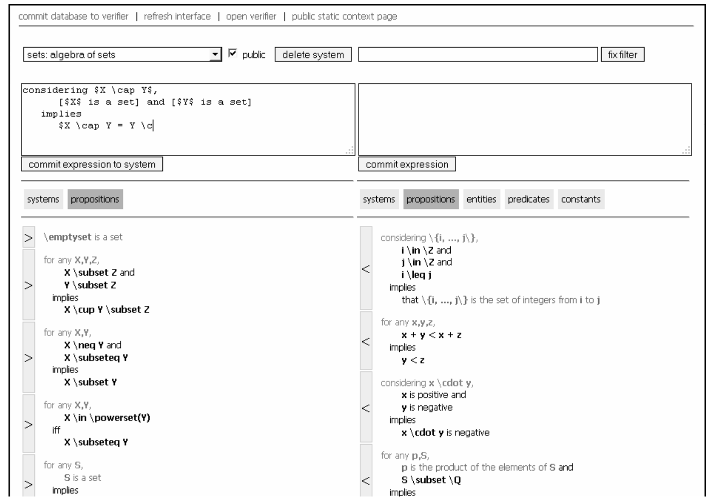
enrolled student
- uses environment to complete assignments



course instructor
- authors lecture notes w/ examples
- assembles assignments
- specifies available propositions

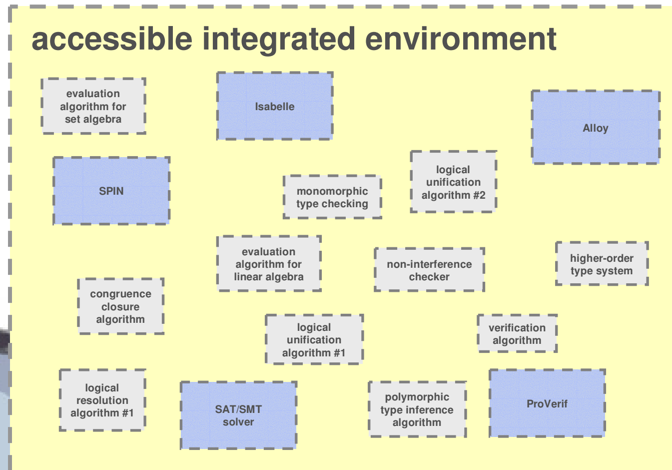
- in earlier work, interfaces were built for collaboratively assembling and organizing a database of propositions written in a familiar syntax



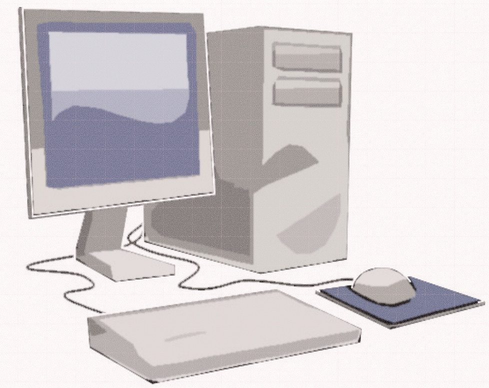


- ongoing work involves building extensions to content management systems (e.g., Drupal, MediaWiki) to support:
  - assembly of lecture notes that include verifiable formal arguments and assignments to be completed in the environment
  - assembly of a database of formal facts written in a familiar syntax
    - grouping of formal facts into collections
    - association of collections of facts with assignments and examples in the notes
  - logging of usage patterns

we describe some of the infrastructure components supporting the tasks in which a formal systems expert may need to engage to implement an environment



enrolled student
- uses environment to complete assignments

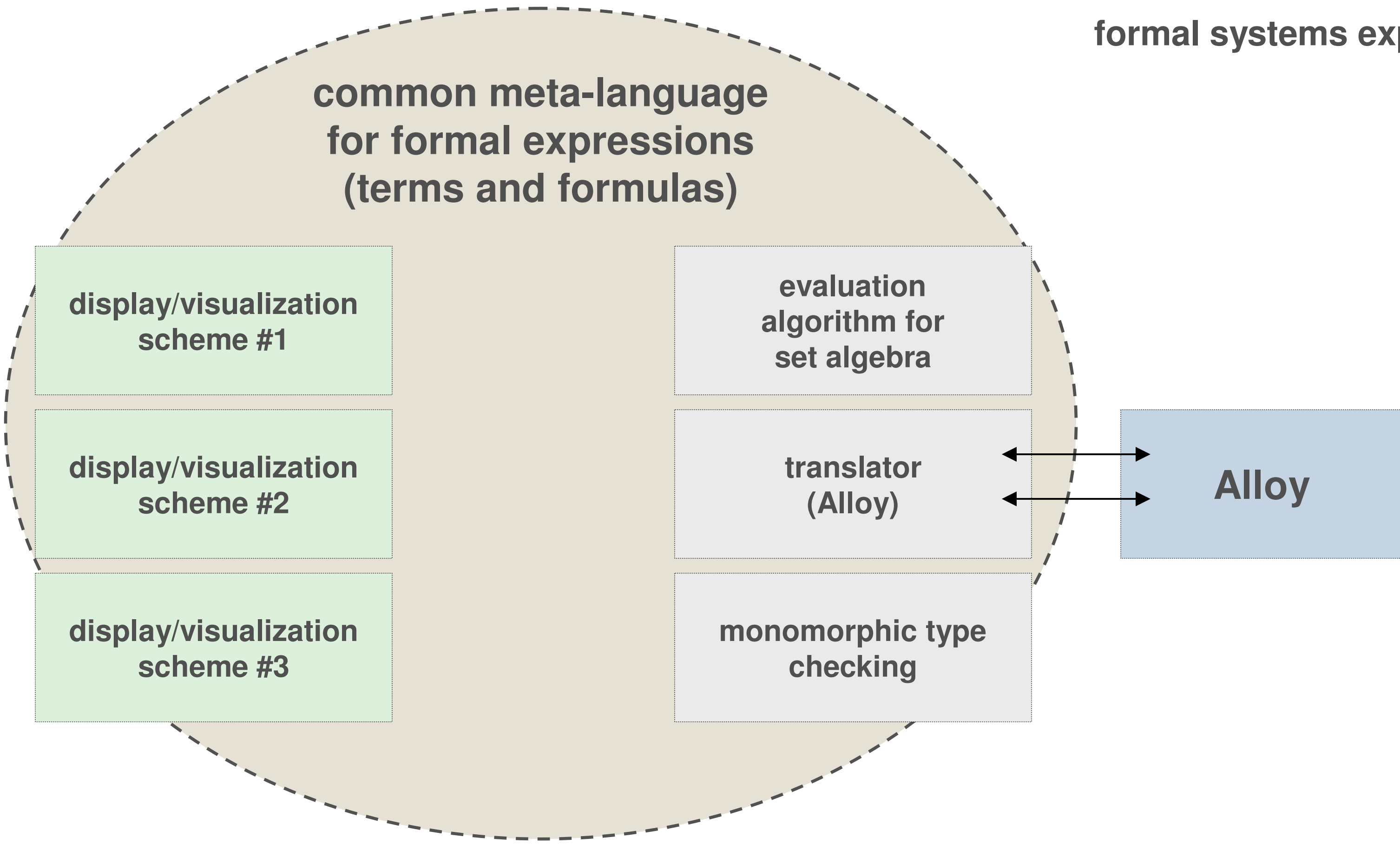


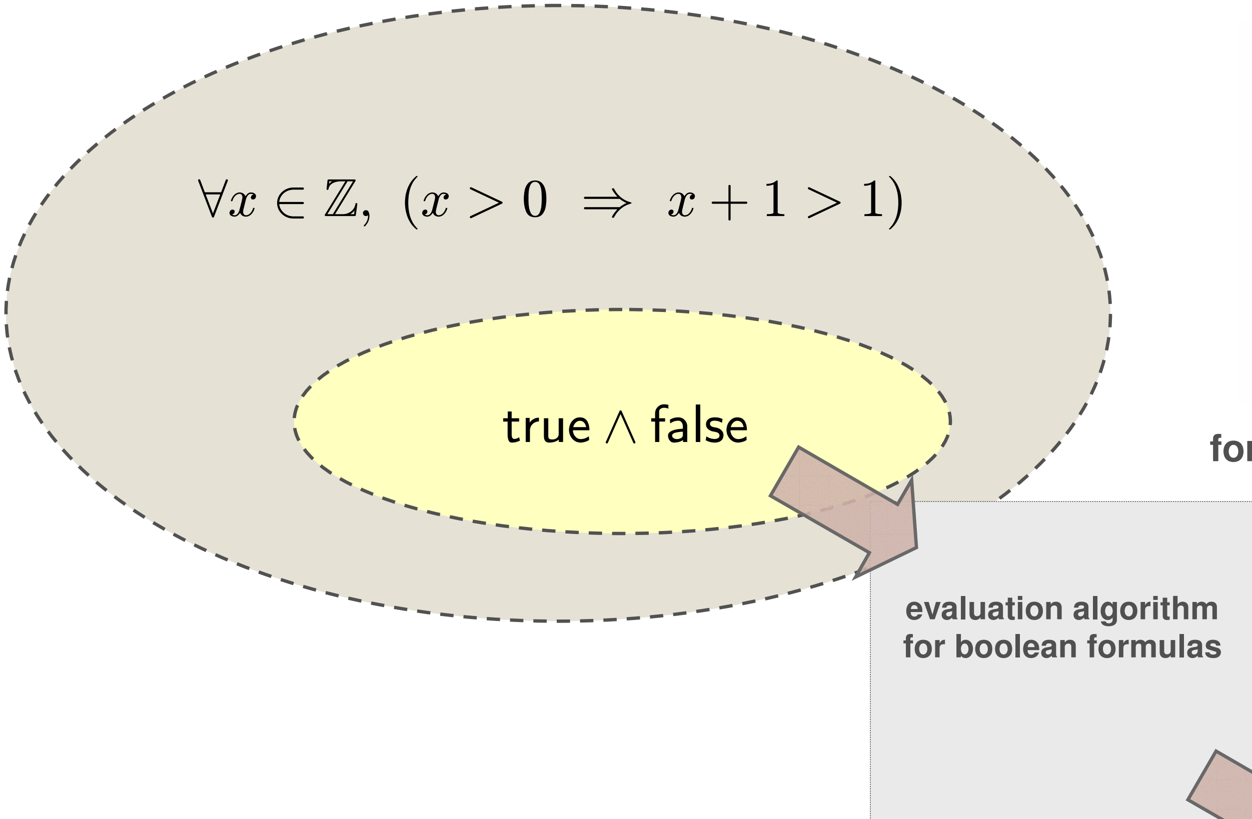
formal systems expert
- implements integrated algorithms
- implements translations to other systems

course instructor
- authors lecture notes w/ examples
- assembles assignments
- specifies available propositions

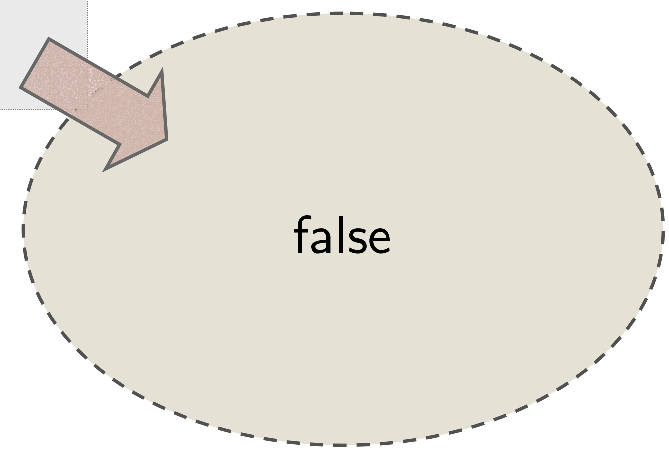
all component algorithms are implemented to process an expression language chosen for the particular application domain supported by the environment

formal systems expert





formal systems expert



components assumed to be expression transformations

- other kinds of algorithms may exist, but are not yet supported

$$\forall x \in \mathbb{Z},\ (x > 0 \ \Rightarrow\ x + 1 > 1)$$

true $\wedge$ false

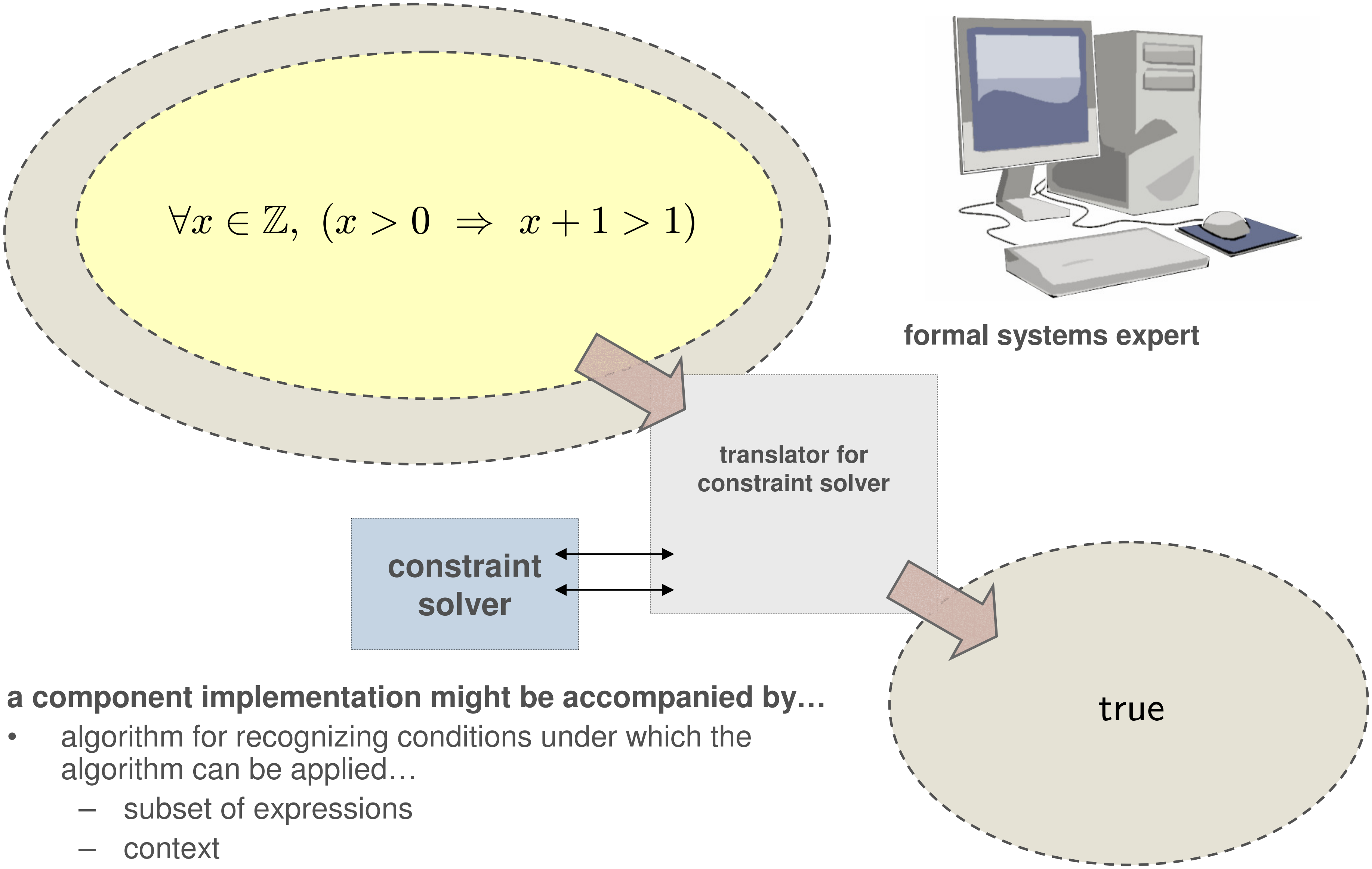
formal systems expert

evaluation algorithm for boolean formulas

false

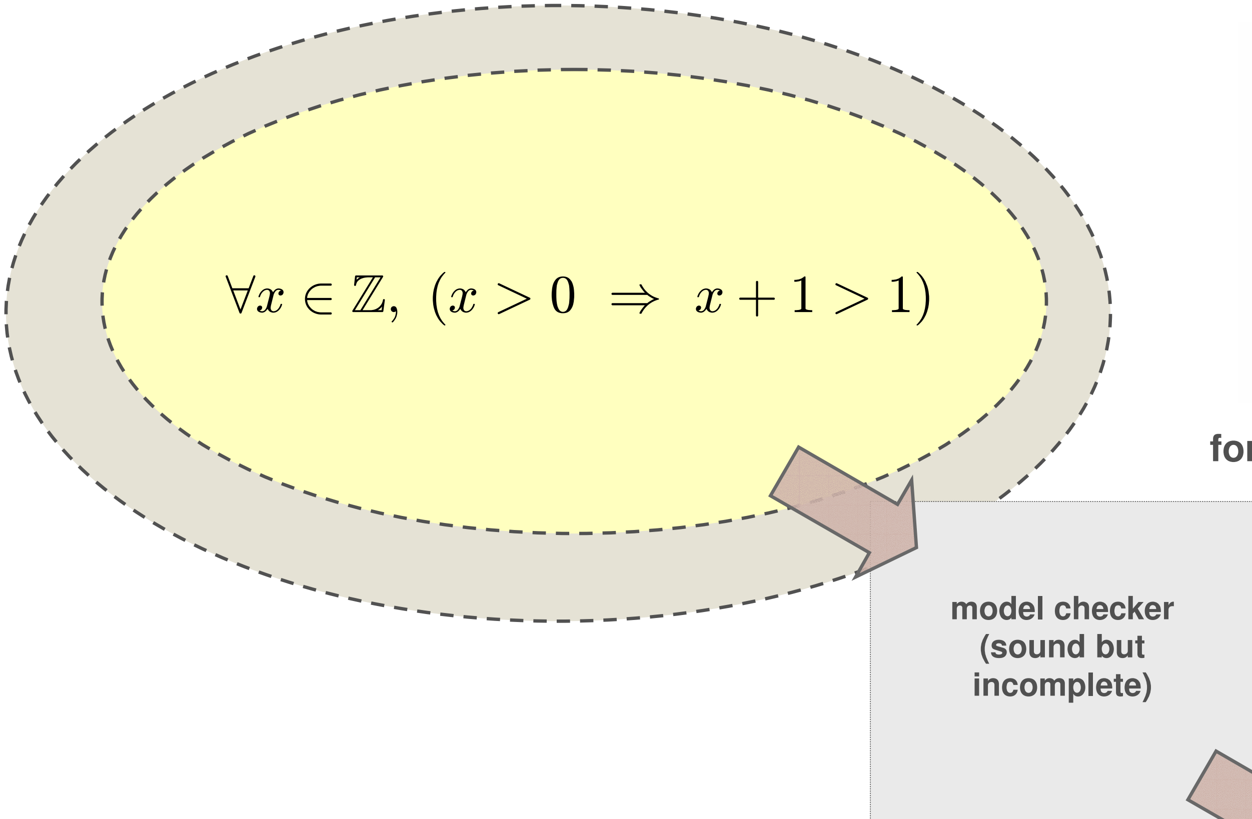
a component implementation might be accompanied by…

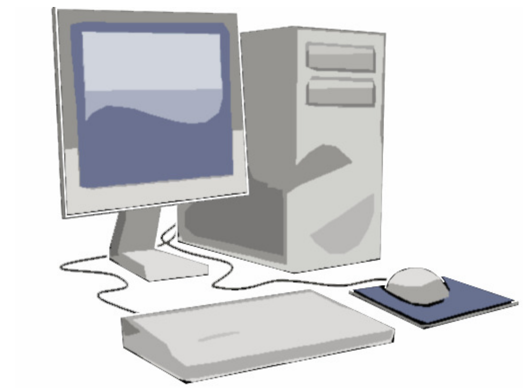
- algorithm for recognizing conditions under which the algorithm can be applied…
  - subset of expressions
  - context
- algorithm for computing upper bound on execution time of the algorithm for an input



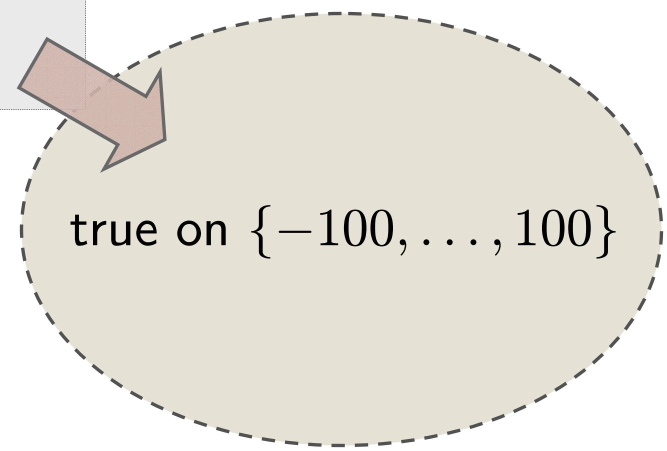
a component implementation might be accompanied by…

- algorithm for recognizing conditions under which the algorithm can be applied…
  - subset of expressions
  - context





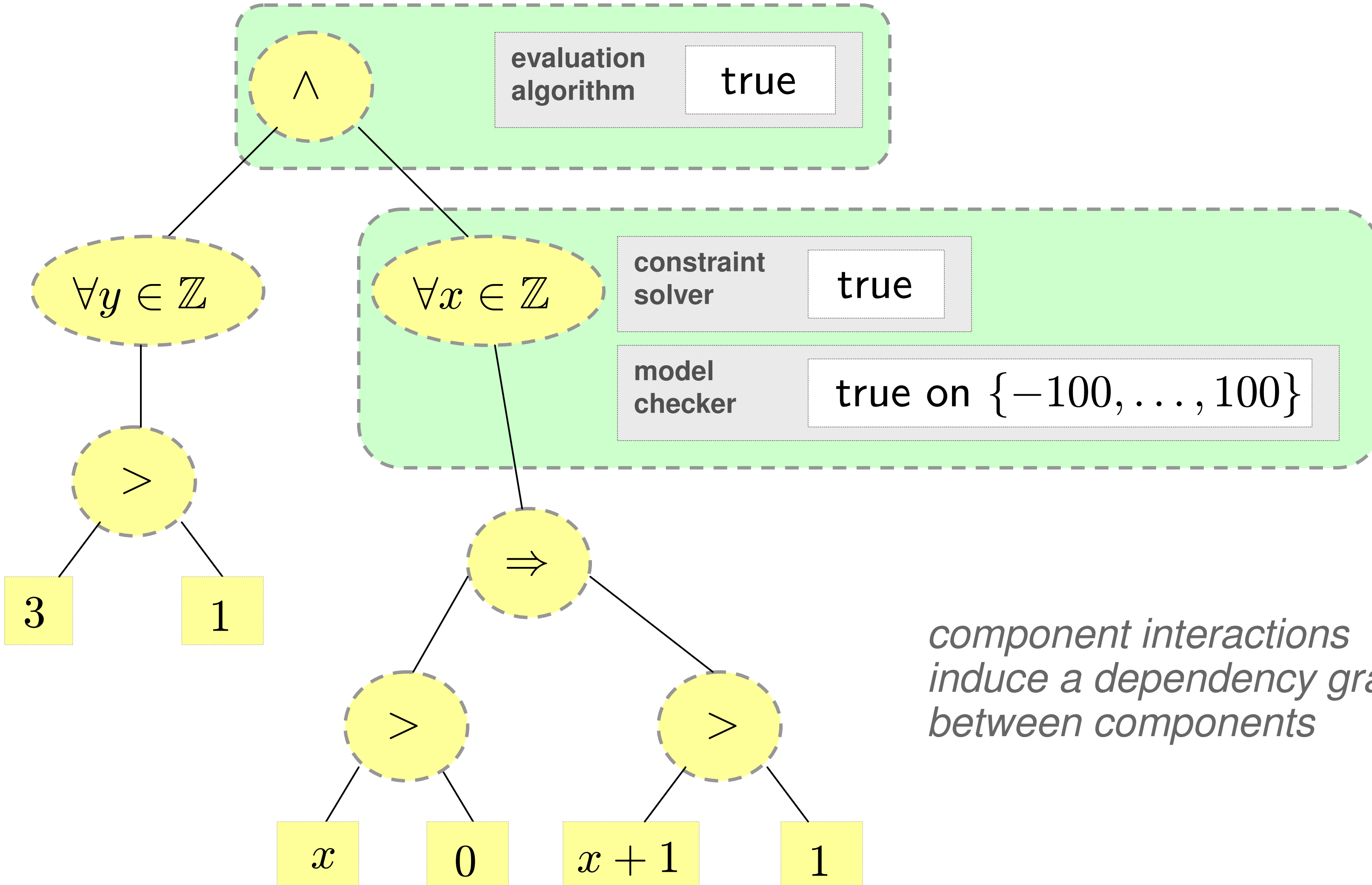
formal systems expert



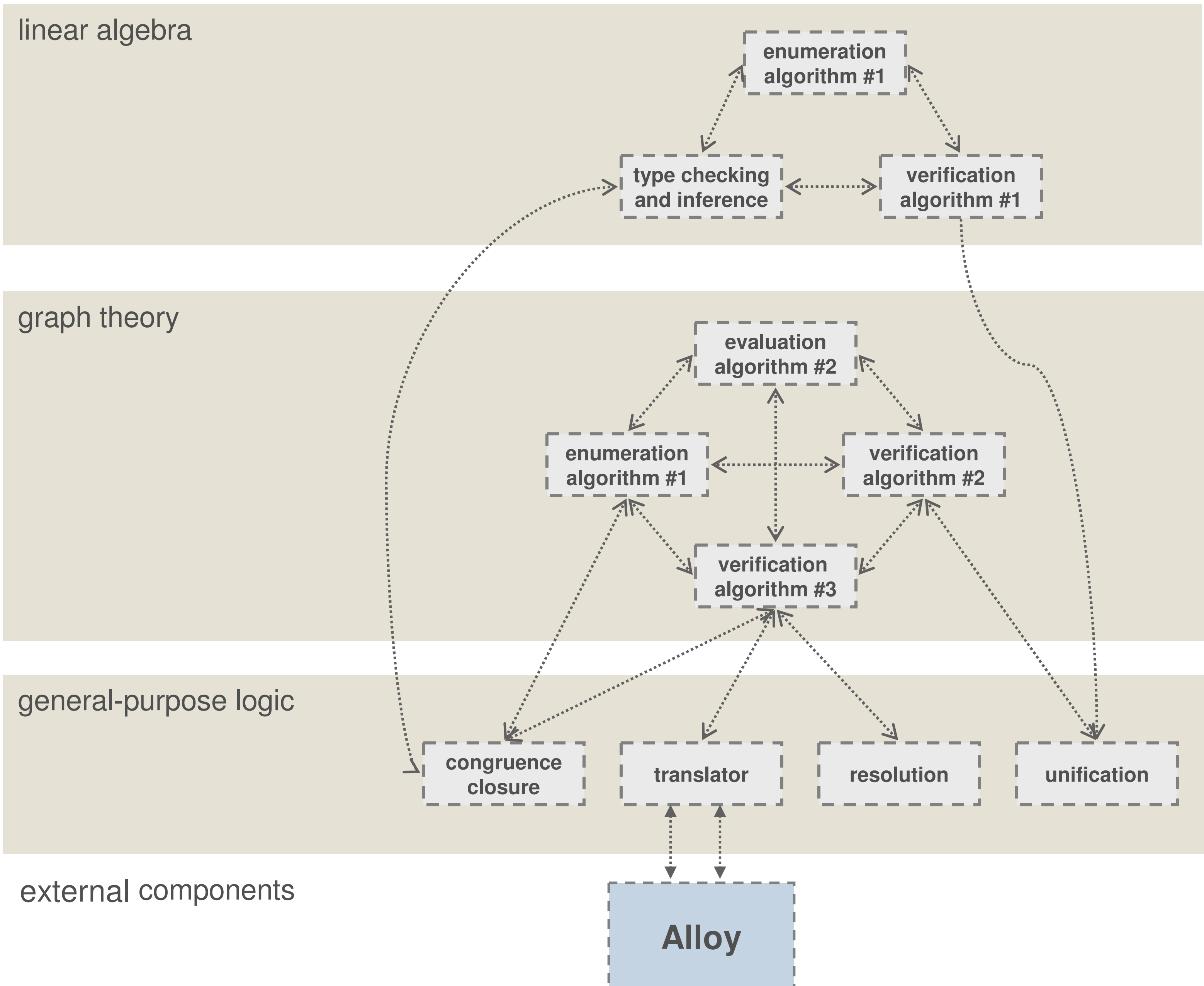
a component implementation might be accompanied by…

- algorithm for recognizing conditions under which the algorithm can be applied…
  - subset of expressions
  - context
- algorithm for generating human-friendly interpretation of its result

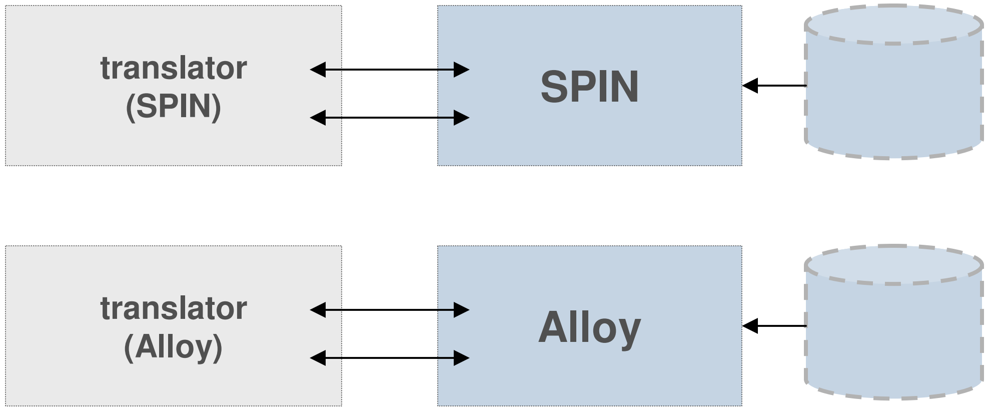
$$(\forall y \in \mathbb{Z},\ 3 > 1) \wedge (\forall x \in \mathbb{Z},\ (x > 0 \ \Rightarrow\ x + 1 > 1))$$

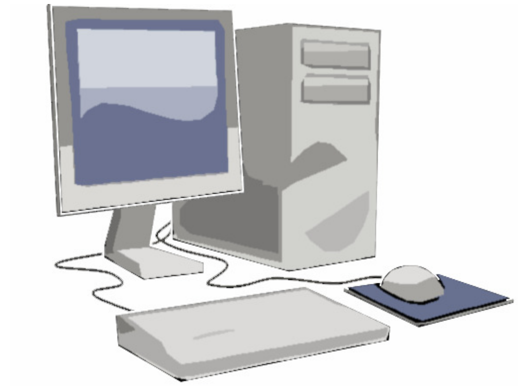


component interactions induce a dependency graph between components

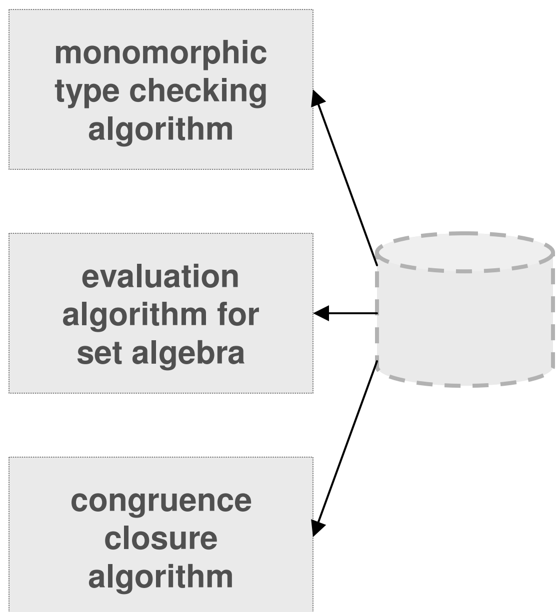

linear algebra
enumeration algorithm #1
type checking and inference
verification algorithm #1
graph theory
evaluation algorithm #2
enumeration algorithm #1
verification algorithm #2
verification algorithm #3
general-purpose logic
congruence closure
translator
resolution
unification
external components
Alloy

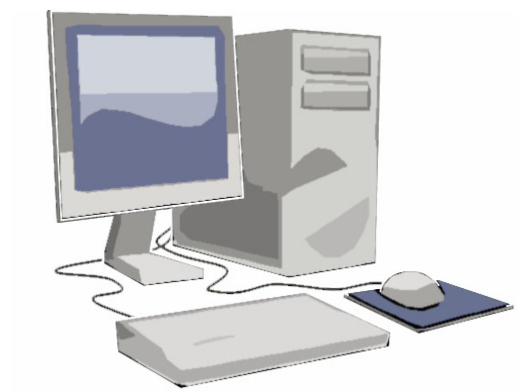
- what are some ways to address the dependencies between components?
- if the dependency graph between components is acyclic (i.e., a DAG)…
  - at compile time, determine dependencies from implementation and generate environment code appropriately
- if the dependency graph has cycles…
  - generate code that continues making passes until convergence
  - generate code that is restricted to, or allows a finite number of passes
    - determined by the user?





formal systems expert

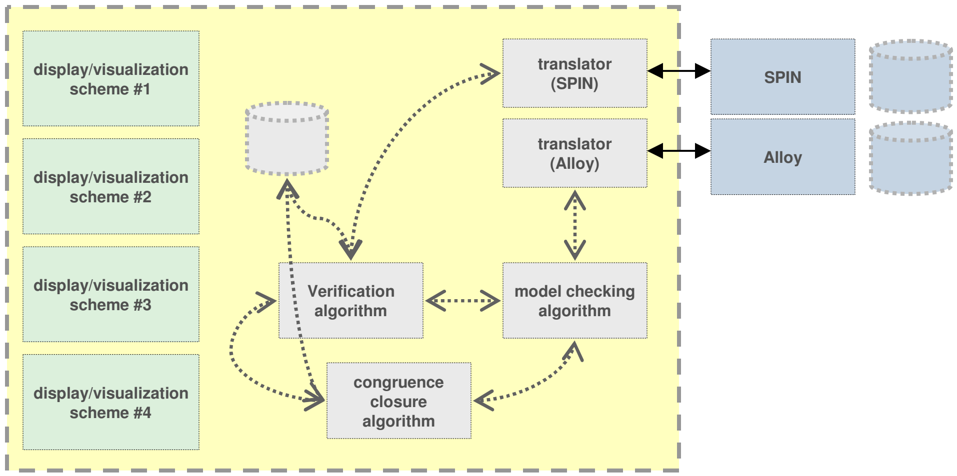


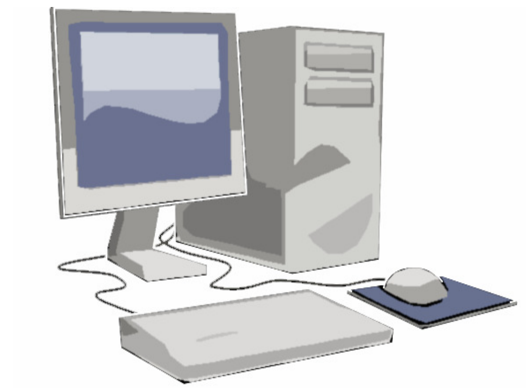


administrator/domain expert

databases contain…

- syntactic idioms
  - constructs
  - predicates
  - operators
- libraries
  - definitions
  - propositions

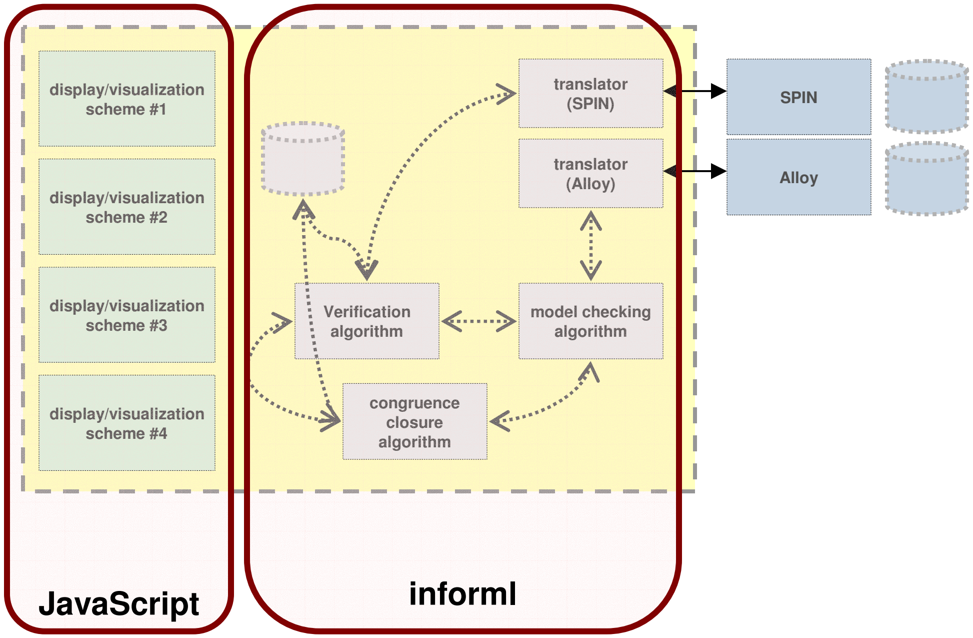


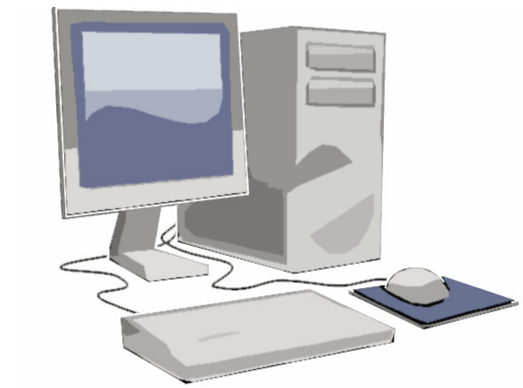


formal systems expert

– the definition of an integrated environment incorporates component algorithms, backend tools, and user interface components

administrator/domain expert

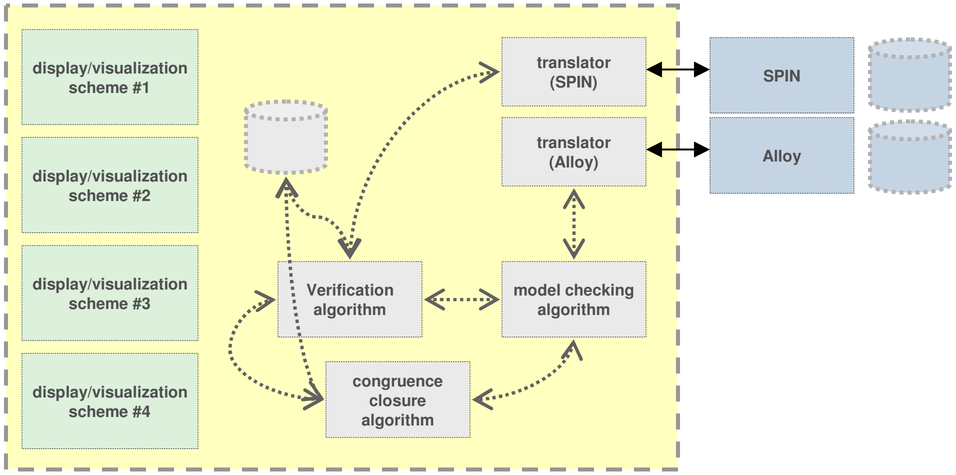


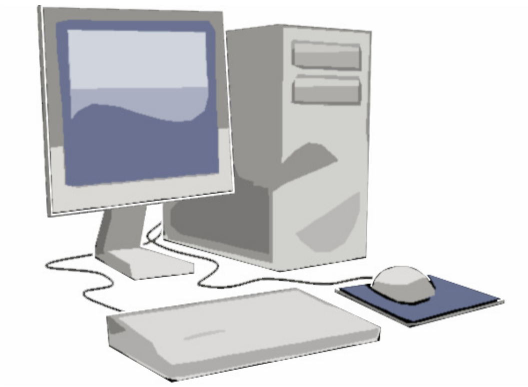


formal systems expert

- the definition of an integrated environment incorporates component algorithms, backend tools, and user interface components
- a custom high-level programming language, **informl**, is used to implement component algorithms
- language features include:
  - easy construction of parsers
  - abstract syntax (i.e., algebraic data types) and supported operations
  - can be compiled to JavaScript, PHP, and Haskell

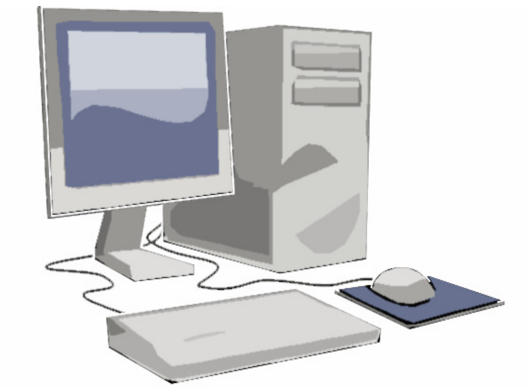
administrator/domain expert



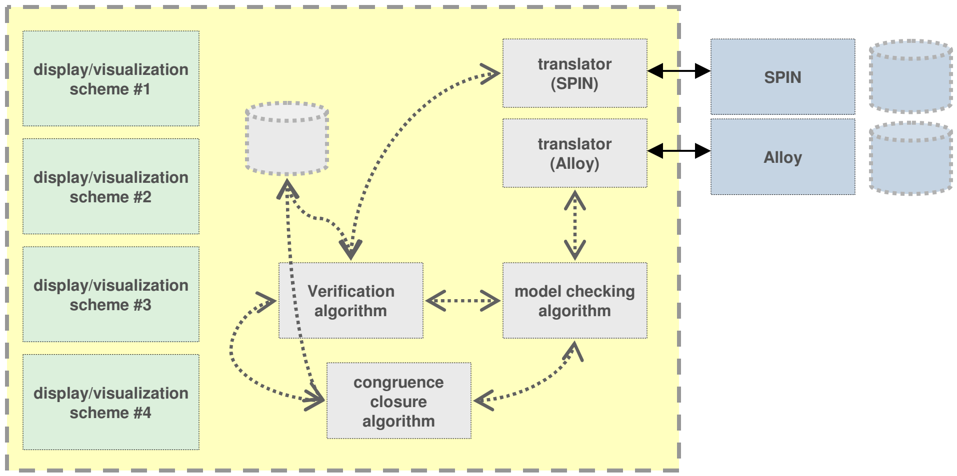


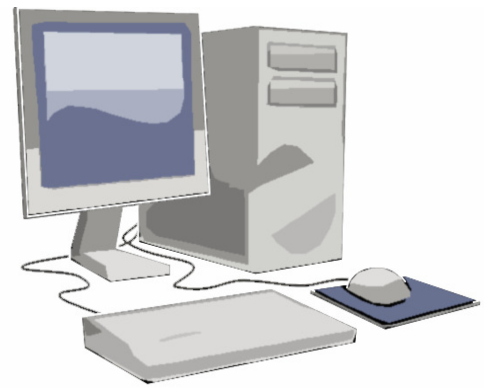
formal systems expert

end-user

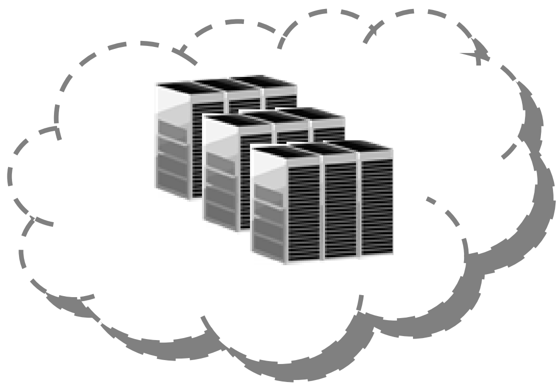


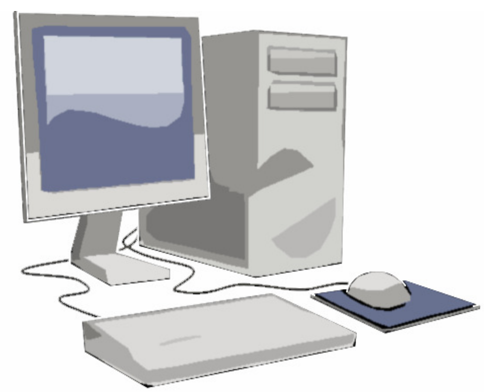
administrator/domain expert





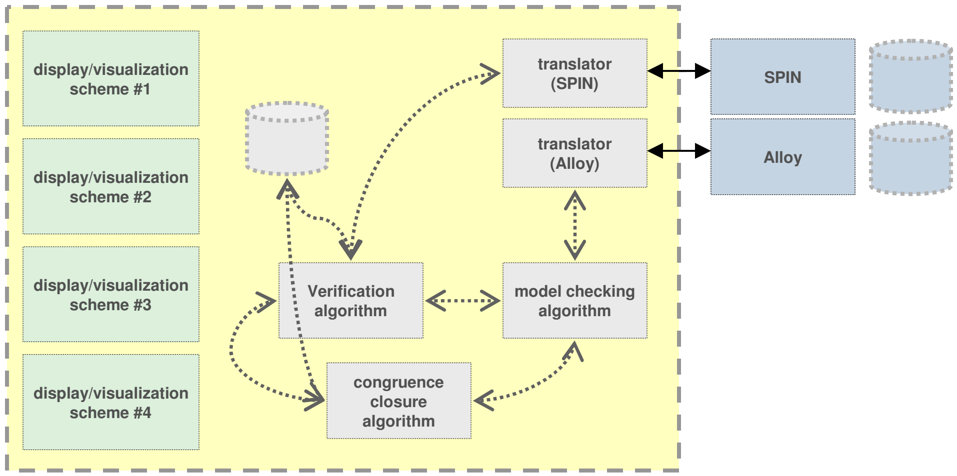
formal systems expert

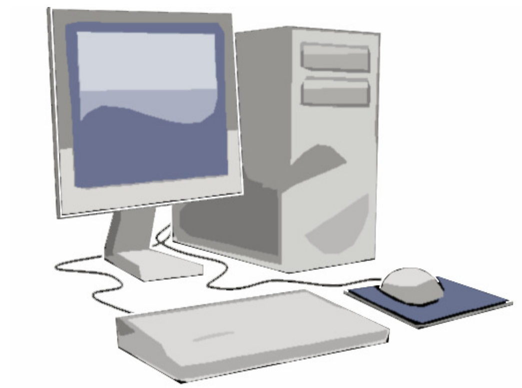




administrator/domain expert

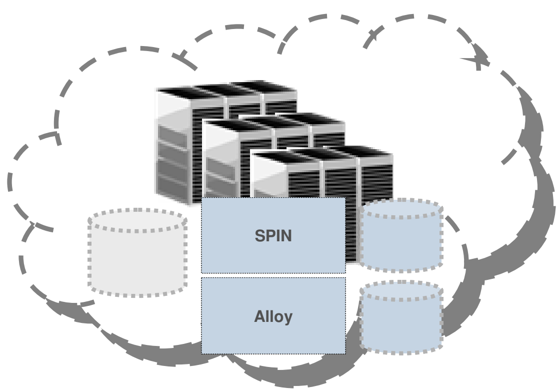
end-user



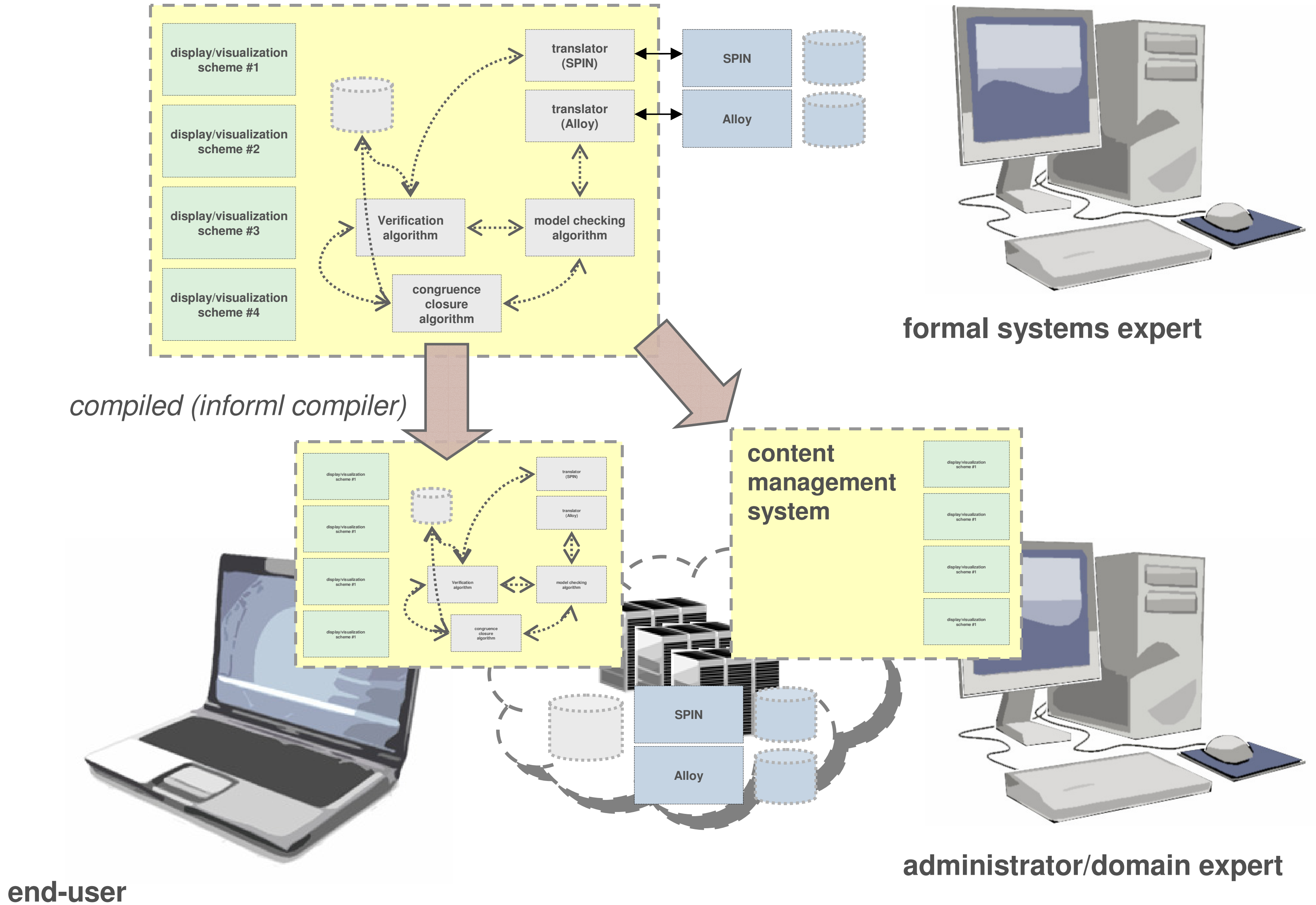


formal systems expert

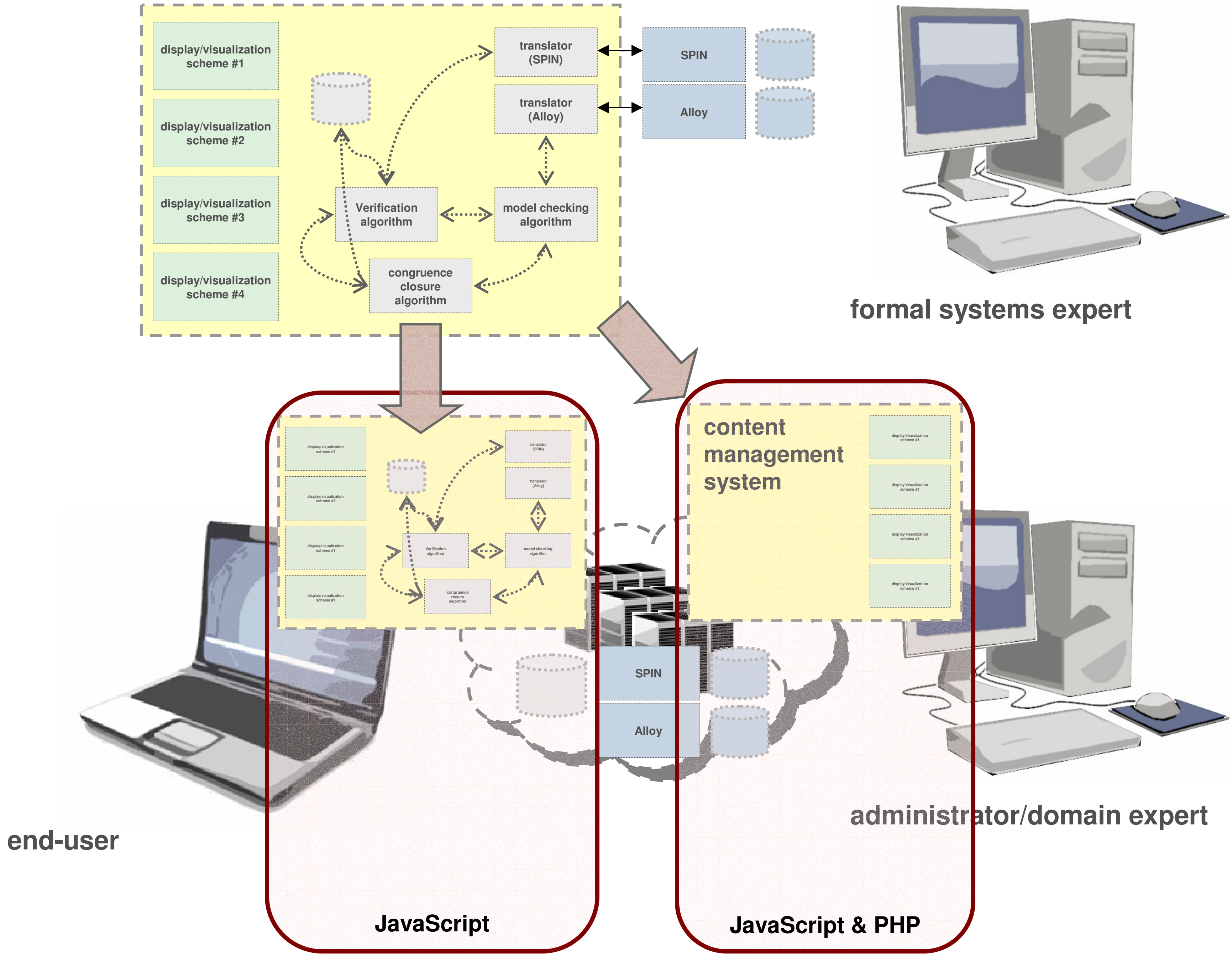
end-user

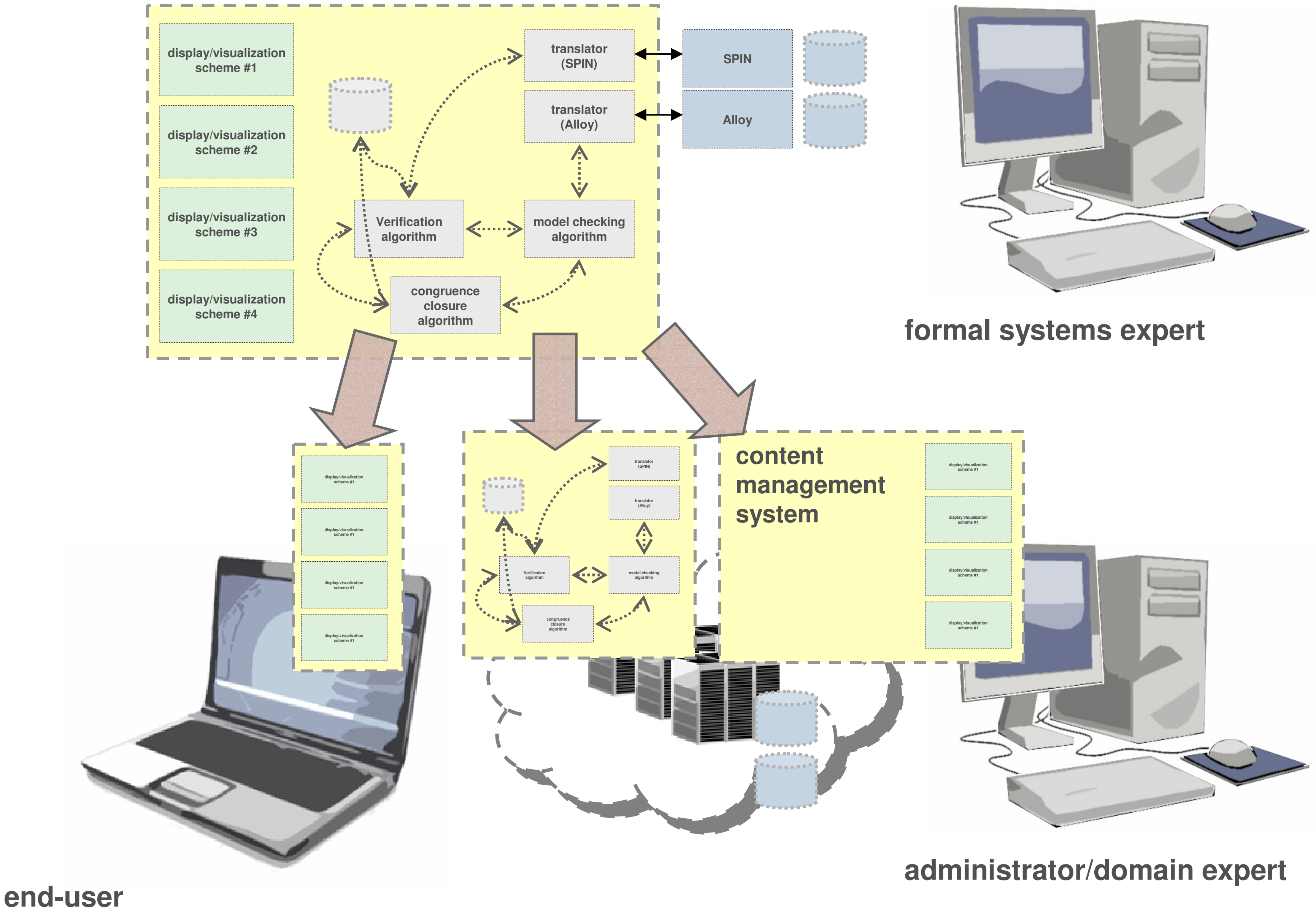


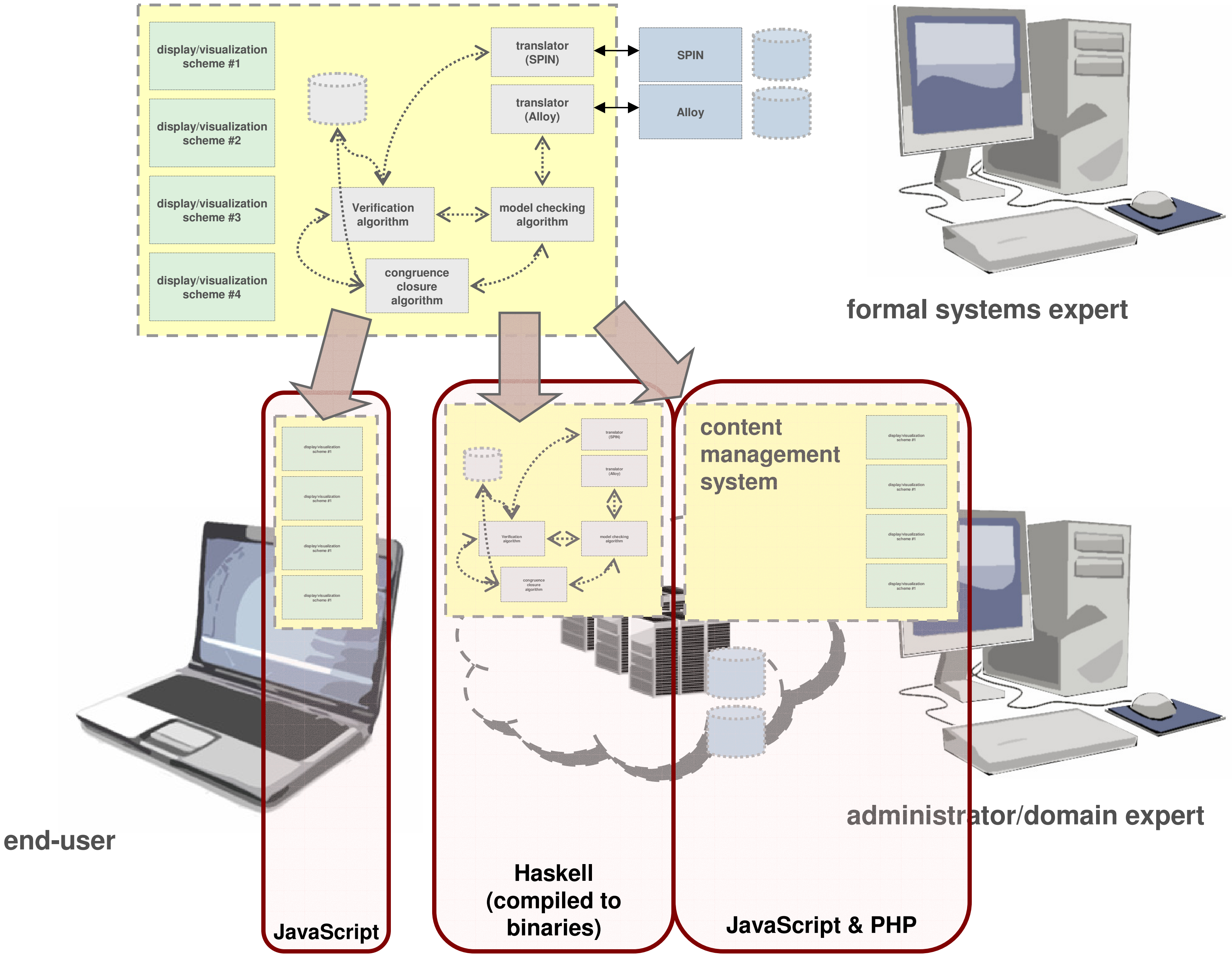
administrator/domain expert

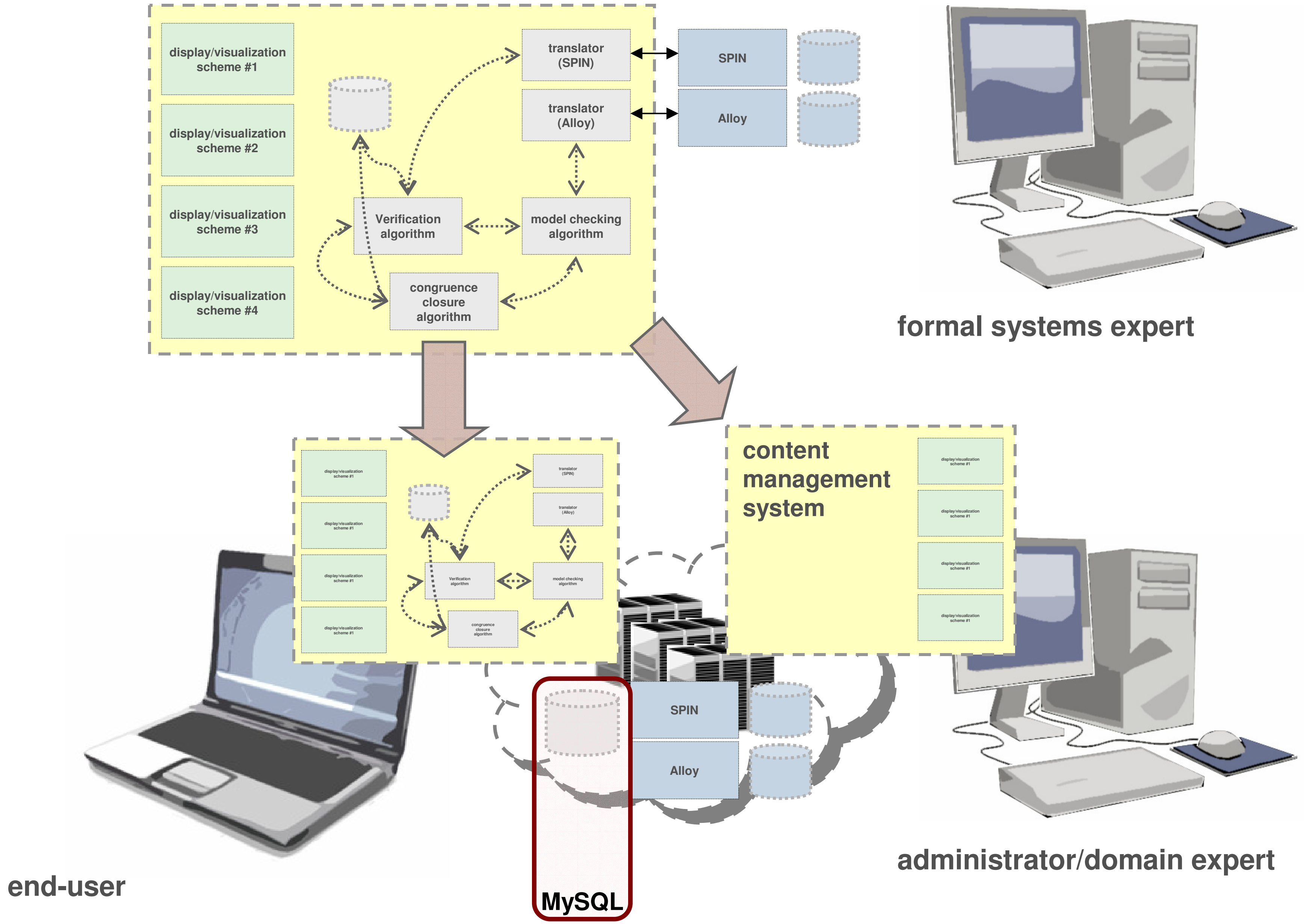


display/visualization scheme #1
display/visualization scheme #2
display/visualization scheme #3
display/visualization scheme #4
translator (SPIN)
translator (Alloy)
SPIN
Alloy
Verification algorithm
model checking algorithm
congruence closure algorithm
formal systems expert
compiled (informl compiler)
content management system
SPIN
Alloy
end-user
administrator/domain expert


display/visualization scheme #1
display/visualization scheme #2
display/visualization scheme #3
display/visualization scheme #4
translator (SPIN)
translator (Alloy)
SPIN
Alloy
Verification algorithm
model checking algorithm
congruence closure algorithm
formal systems expert
content management system
SPIN
Alloy
end-user
administrator/domain expert
JavaScript
JavaScript & PHP


display/visualization scheme #1
display/visualization scheme #2
display/visualization scheme #3
display/visualization scheme #4
translator (SPIN)
translator (Alloy)
SPIN
Alloy
Verification algorithm
model checking algorithm
congruence closure algorithm
formal systems expert
content management system
end-user
administrator/domain expert


display/visualization scheme #1
display/visualization scheme #2
display/visualization scheme #3
display/visualization scheme #4
translator (SPIN)
translator (Alloy)
SPIN
Alloy
Verification algorithm
model checking algorithm
congruence closure algorithm
formal systems expert
content management system
end-user
administrator/domain expert
JavaScript
Haskell (compiled to binaries)
JavaScript & PHP


display/visualization scheme #1
display/visualization scheme #2
display/visualization scheme #3
display/visualization scheme #4
translator (SPIN)
translator (Alloy)
SPIN
Alloy
Verification algorithm
model checking algorithm
congruence closure algorithm
formal systems expert
content management system
SPIN
Alloy
MySQL
end-user
administrator/domain expert

- use case: prototype used in classroom instruction
  - introductory linear algebra course
    - primarily undergraduates; about 75% sophomores or freshmen
  - integrated environment utilized…
    - by instructor to present examples integrated into notes
    - by students to complete homework assignments
    - by graders
  - integrated components include
    - congruence closure computation
    - monomorphic type checking
    - limited first-order logical verification
    - set algebra and linear algebra evaluation algorithms
- use case: secure network protocols (other ongoing work)
  - integration of non-interference checking, congruence closure computation, type checking, Alloy, and SPIN
- open question: what is a meaningful way to evaluate the effectiveness of an integrated environment?
  - surveys, student performance, etc.
  - are there useful techniques in other disciplines?

acknowledgements

Andrei Lapets
Rick Skowyra
Christine Bassem
Nate Soule

Assaf Kfoury
Azer Bestavros

Computer Science Dept., Boston University
Hariri Institute at Boston University
NSF Grants No. 0820138 and No. 0720604