CAS 734 Winter 2006

06 The Little Theories Method

William M. Farmer

Department of Computing and Software McMaster University

5 December 2006

Little Theories Method

- A complex body of mathematical knowledge is represented as a network of theories.
  - Bigger theories are composed of smaller theories.
  - Theories are linked by interpretations.
  - Both knowledge and reasoning are distributed over the network.
- Benefits:
  - Mathematics can be developed in the right language at the right level of abstraction.
  - Emphasizes reuse: if A is a theorem of T, then A may be reused in any "instance" of T.
  - Enables perspective switching.
  - Enables parallel development.
  - Possible inconsistency can be isolated.

Theory Interpretations

- A translation Φ from T to T' is a function that maps the primitive symbols of T to expressions of T' satisfying certain syntactic conditions.
- Φ determines:
  - A mapping of expressions of T to expressions of T'.
  - A set of sentences called obligations.
- Φ is an interpretation if it maps the theorems of T to theorems of T'.
  - Sufficient condition: all the obligations of Φ are theorems of T'.
- An interpretation is faithful if it also maps the nontheorems of T to nontheorems of T'.
  - A faithful interpretation is a generalization of a conservative extension.
- Interpretations are information conduits!

Uses of Theory Interpretation

- In logic:
  - Showing properties of theories, e.g., decidability.
  - Comparing the "strength" of theories.
- In computer science:
  - Formalizing the notion of refinement.
  - Module instantiation.
  - Computational complexity.
- In informal and formalized mathematics:
  - Organization of mathematical knowledge.
  - Reuse of theorems, definitions, computations, etc.
  - Problem transference.
  - Trustable communication between mathematical systems.

Applications of the Little Theories Method

- Software specification systems.

  Examples: EHDM, IOTA, KIDS, OBJ, Specware.

- Theorem proving systems.

  Examples: Coq, Ergo, IMPS, Isabelle, PVS.

- Computer algebra systems.

  Example: Axiom.

Example: Theory of Computer Networks

- Theory name: Networks.
- Language: A language of many-sorted first-order logic with the following sorts and function symbols:

Sorts	Function symbols
boxes	box-of-interface
wires	wire-of-interface
interfaces	address-of-interface
addresses	

- Example axioms:
  - "Every box has a unique loopback interface".
  - "The address of a loopback interface is 127.0.0.1".

Example: Theory of Bipartite Graphs

- Theory name: Bipartite Graphs.
- Language: A language of many-sorted first-order logic with the following sorts and function symbols:

Sorts	Operators
red-nodes	red-node-of-edge
blue-nodes	blue-node-of-edge
edges	

- No explicit axioms.

Example: Bipartite Graphs to Networks

- Let $\Phi_{BG \to N}$ be the translation from Bipartite Graphs to Networks defined by:

  red-nodes $\mapsto$ boxes.
  blue-nodes $\mapsto$ wires.
  edges $\mapsto$ interfaces.
  red-node-of-edge $\mapsto$ box-of-interface.
  blue-node-of-edge $\mapsto$ wire-of-interface.

- $\Phi_{BG \to N}$ has no obligations.
- $\Phi_{BG \to N}$ is an interpretation.
  - "Transitivity of red-to-red connectivity" maps to "transitivity of box-to-box connectivity".

Example: Symmetry Interpretation

- Let $\Phi_{BG \to BG}$ be the translation from Bipartite Graphs to Bipartite Graphs defined by:

  red-nodes $\mapsto$ blue-nodes.
  blue-nodes $\mapsto$ red-nodes.
  edges $\mapsto$ edges.
  red-node-of-edge $\mapsto$ blue-node-of-edge.
  blue-node-of-edge $\mapsto$ red-node-of-edge.

- $\Phi_{BG \to BG}$ has no obligations.
- $\Phi_{BG \to BG}$ is an interpretation.
  - "Transitivity of red to red connectivity" maps to "transitivity of blue to blue connectivity".

Theory Interpretations in IMPS

- A translation Φ from T_1 to T_2 is a pair of functions

  $$\mu : \text{atomic-sorts}(T_1) \rightarrow \text{sorts}(T_2) \cup \text{sets}(T_2) \cup \text{unary-predicates}(T_2)$$
  $$\nu : \text{constants}(T_1) \rightarrow \text{expressions}(T_2)$$

  satisfying certain syntactic conditions.

- Φ determines:
  - A mapping from expressions(T_1) to expressions(T_2) that is a homomorphism with respect to the logic's expression constructors.
  - A set of sentences called obligations is derived from the axioms and sorting information of T_1.
- Φ is an interpretation of T_1 in T_2 if, for each sentence A, $T_1 \models A$ implies $T_2 \models \Phi(A)$.

Basic Theorems

Interpretation Theorem. Φ is an interpretation if each of its obligations is valid in T_2.

Relative Satisfiability. If Φ is an interpretation and T_2 is satisfiable, then T_1 is satisfiable.

Uses of Interpretations in IMPS

- Reuse of theorems and definitions.
- Isolation of possible inconsistency.
- Theory instantiation.
- Polymorphic operators.
- Symmetry and duality proofs.
- Theory ensembles.
- Theorem finding via translation matching.

Partial Orders

```
(def-language LANGUAGE-FOR-PARTIAL-ORDER
  (base-types uu)
  (constants (prec "[uu,uu -> prop]")))

(def-theory PARTIAL-ORDER
  (language language-for-partial-order)
  (component-theories h-o-real-arithmetic)
  (axioms
   (prec-transitivity
    "forall(a,b,c:uu, a prec b and b prec c implies a prec c)")
   (prec-reflexivity
    "forall(a:uu, a prec a)")
   (prec-anti-symmetry
    "forall(a,b:uu, a prec b and b prec a implies a = b)")))

(def-constant REV%PREC
  "lambda(a,b:uu, b prec a)"
  (theory partial-order))
```

Partial Orders Symmetry

```
(def-translation ORDER-REVERSE
  (source partial-order)
  (target partial-order)
  (fixed-theories h-o-real-arithmetic)
  (constant-pairs
    (prec rev%prec)
    (rev%prec prec))
  (theory-interpretation-check using-simplification))

(def-renamer FIRST-RENAMER
  (pairs (prec%increasing rev%prec%increasing)
         (prec%majorizes prec%minorizes)
         (prec%sup prec%inf)))

(def-transported-symbols
  (prec%increasing prec%majorizes prec%sup)
  (translation order-reverse)
  (renamer first-renamer))
```

Metric Spaces

```
(def-language METRIC-SPACES-LANGUAGE
  (embedded-languages h-o-real-arithmetic)
  (base-types pp)
  (constants
   (dist "[pp,pp -> rr]")))

(def-theory METRIC-SPACES
  (component-theories h-o-real-arithmetic)
  (language metric-spaces-language)
  (axioms
   (positivity-of-distance
    "forall(x,y:pp, 0<=dist(x,y))" transportable-macete)
   (point-separation-for-distance
    "forall(x,y:pp, x=y iff dist(x,y)=0)" transportable-macete)
   (symmetry-of-distance
    "forall(x,y:pp, dist(x,y) = dist(y,x))" transportable-macete)
   (triangle-inequality-for-distance
    "forall(x,y,z:pp, dist(x,z)<=dist(x,y)+dist(y,z))"
    transportable-macete)))
```

Metric Spaces Theory Ensemble

```
(def-theory-ensemble METRIC-SPACES)

(def-constant OPEN
  "lambda(o:sets[pp], forall(x:pp,
     x in o
      implies
     forsome(delta:rr,0<delta and ball(x,delta) subseteq o)))"
 (theory metric-spaces)
 (usages transportable-macete))

(def-theory-ensemble-multiple metric-spaces 2)

(def-theory-ensemble-overloadings metric-spaces (1))

(def-constant CONTINUOUS
  "lambda(f:[pp_0 -> pp_1],
     forall(v:sets[pp_1], open(v) implies open(inv_image(f,v))))"
  (theory metric-spaces-2-tuples))
```

Group Actions

```
(def-language GROUP-ACTION-LANGUAGE
  (embedded-language groups-language)
  (base-types uu)
  (constants
   (act "[uu,gg -> uu]")))

(def-theory GROUP-ACTIONS
  (language group-action-language)
  (component-theories groups)
  (axioms
   (act-id
    "forall(alpha:uu,g:gg, act(alpha,e) = alpha)"
    rewrite transportable-macete)
   (act-associativity
    "forall(alpha:uu,g,h:gg,
       act(alpha,g mul h) = act(act(alpha,g),h))"
    transportable-macete)))
```

Group Actions Interpretations

```
(def-translation ACT->RIGHT-MUL
  (source group-actions)
  (target groups)
  (fixed-theories h-o-real-arithmetic)
  (sort-pairs
   (uu "gg"))
  (constant-pairs
   (act "mul"))
  (theory-interpretation-check using-simplification))

(def-translation ACT->SET%CONJUGATE
  (source group-actions)
  (target groups)
  (fixed-theories h-o-real-arithmetic)
  (sort-pairs
   (uu "sets[gg]"))
  (constant-pairs
   (act "set%conjugate"))
  (theory-interpretation-check using-simplification))
```

Translation Matching

- Let E_i be an expression of theory T_i for $i = 1, 2$.
  - E_1 is the pattern.
  - E_2 is be "matched" against E_1.
- E_2 translation matches E_1 if there is an interpretation Φ from T_1 to T_2 and a substitution σ such that

$$E_2 = \sigma(\Phi(E_1)).$$

- Φ "instantiates" the primitive symbols in E_1.
- σ instantiates the variables in $\Phi(E_1)$.

Translation Matching Algorithm

- The interpretation is computed as follows:
  1. From E_1 and E_2 generate a "partial translation" $\widetilde{\Phi}$ from T_1 to T_2.
  2. Find an interpretation Φ from a supertheory of T_1 to a subtheory of T_2 that extends $\widetilde{\Phi}$.
  3. If step 2 fails, check to see if $\widetilde{\Phi}$ is an interpretation from T_1 to T_2.
- The substitution is computed in a routine fashion.

Example: Translation Matching

- Let E_1 be red-node-of-edge$(e) = r$ where e : edges and r : red-nodes.
- Let E_2 be box-of-interface(I-34) = Sun-78.
- Does E_2 translation match E_1?
- The partial translation is $\widetilde{\Phi}$ where:

  red-nodes $\mapsto$ boxes.
  blue-nodes $\mapsto$?
  edges $\mapsto$ interfaces.
  red-node-of-edge $\mapsto$ box-of-interface.
  blue-node-of-edge $\mapsto$?
- The interpretation is $\Phi_{\mathrm{BG}\to\mathrm{N}}$ that extends $\widetilde{\Phi}$.
- The substitution maps e to I-34 and r to Sun-78.

Trustable Communication between Mathematical Systems

- Basic scheme for trustable communication:
  - Each system has one or more interfaces that provide services to other systems.
  - A connection from an interface I_1 to an interface I_2 is an asymmetric pair $C = (\text{export}, \text{import})$ of translations such that import is an interpretation.
  - A request for a service θ provided by I_2 is sent from I_1 to I_2 via the export translation, and an answer to the request is sent back from I_2 to I_1 via the import interpretation.
- The answer received by interface I_1 is correct provided:
  - The answer produced by the service θ is a theorem of the underlying theory of I_2.
  - The import translation is an interpretation.