WhyMSL

May 18, 2011

Contents

1 ConjectureOp

```
module ConjectureOp

definitions

types

  bla = token;

end ConjectureOp
```

Function or operation	Coverage	Calls
ConjectureOp	0.0%	0
ConjectureOp.vdmsl	0.0%	0

2 WhyM

```
/*
-----------------
08/05/2011 (Leo) - first version based on Cliff's BBN 1749 (from 06/05/2011)
10/05/2011 (Leo) - copy-paste into "WhyMCliff" + update notions after discussions with Andrius
12/05/2011 (Leo) - meeting with Cliff for discussion: explained up to Evidence
13/05/2011 (Leo) - froze WhyM0 + added notions of Proof/Disproof/Gap/Attempt etc in detail
-----------------
*/
```

```
module WhyM

definitions

types

------------------------------------------------------------------------------
--- Cliff's "world"
---  * AI4FM strategy language

 Sigma :: map Id to Theory;  -- state for models of why operations

------------------------------------------------------------------------------
--- Theory
---  * Theory records intent information and inference
---  * link with underlying log data, here just a pointer
---  * hints are sequence of whys, and represent positive knowledge
---  * clues are instantiate whys, yet they are just pieces of the jigsaw
---  * strategies are not to be updated, except when/with learning
---    (i.e., they are those created during proof and fixed).
---
---  * should uses/specialised be like "inherited" features? See Isabelle Locale?

 Theory ::
  specialises : Id       -- dependencies down
  log   : Log       -- associated log (pointer)
  uses  : set of Id      -- dependencies up
  ttypes  : set of Id      -- known types
  operators : map Operator to OpDefn  -- tagged/known ops
  results  : map Id to Conjecture   -- proved conjectures
  strategies : set of (Why * Hints)    -- relates why with hints

 -- TODO: also comparative/ordering/ranking notions
 --   between various Theory components / structs
 ;

------------------------------------------------------------------------------
--- Operators
---  * Augumented from tool term structure with meta-tagging?

 Operator ::
  id  : Id
  cat  : OpCategory
  assoc : OpAssoc
  syntax : OpSyntax
  prec : nat;

 OpCategory  = <Relation> | <Function>; --| <Generic>;
 OpSyntax = <Infix> | <Posfix> | <Mixfix> | <Nofix>;
 OpAssoc  = <ASSOC_LEFT> | <ASSOC_RIGHT>; --| <ASSOC_NONE>;

 OpDefn ::
  type: Signature
  defn: Definition;

 -- exrta meta-information given by the user about a defined term
 Definition ::
  term: Term     -- text from tool Term structure
  props: set of DefProp  -- set of properties of interest (e.g., does commute with OpY)   TODO: SHAPE, PLEASE!
  aprops: set of DefProp  -- set of anti-properties of interest (e.g., doesn't commute with OpX)
  concept: Concept   -- what does it define?
  structure: set of Structure -- expected use of definition
  origin: DefOrigin   -- where the definition comes from?
  intent: seq of char   -- user textual description of definition
  related: set of DefLink  -- related terms/definitions/lemmas?
```

```
 witnesses: map Id to Pred -- possible witnesses for all quantified variables of interest?
 inv deft == is not yet specified -- links Pred vars with dom Id from witness
 ;

-- properties of interest for the definition? like assoc/comm/dist etc.
-- what parameters to have, if any?
DefProp = token;

-- what kind of definition is this? useful to scoring mechanisms and strategy choices?
Concept =
   <CORE>  -- concept considered "core" by the user
 | <EXTENSION> -- extension of a core concept
 | <INVARIANT> -- invariant of data structure? +/-
 | <TOY>   -- part of a toy-problem abstraction
 -- TODO: what else? could these be learned/created?
 ;

-- most likely/expected use of definition during a proof
Structure =
   <CONSTRUCTOR> -- constructor
 | <DESTRUCTOR> -- destructor
 | <FUNCTIONAL>  -- function ? unnecessary?
 | <POINTWISE> -- definition is given/used pointwise
 | <SET_BASED> -- definition is for set based resoning
 | <SEQ_BASED> -- definition is for seq/list based reasoning
 | <EXTENSIONAL> -- equality as quantified comparison (e.g., x=y iff (ALL i: x @ i:y) and...)
 | <LIEBINITZ> -- equality as substitution of equals-for-equals
 -- TODO: what else? could these be learned?
 ;

-- definition source: problem itself, or user insight, or other tool/theory
DefOrigin = <EXTENSIONAL> | <INTENTIONAL> | <EXTERNAL>;

-- definition links: other definitions; conjectures; theories
DefLink = Definition | Conjecture | Theory;

-------------------------------------------------------------------------------
--- Conjectures
---  * Augumented from tool term structure with meta-tagging + Evidence
---
---  * justifications map a given proof (Id) to a set of possible proof evidence
---    (i.e., Cliff's proof as either structured / declarative / axiom-use, etc)
---    note this also allows for mixed Proof/Disproof within evidence set (!)
---
---  * a set of clues is given as places where the conjecture might be useful
---  ? what leads to a clue? proof attempt or another conjecture?


-- conjecture as in sequent calculus plus extra model Why info.
-- assuming only "goal" identify a conjecture (used ":-")
Conjecture ::
 hypothesis  :- seq of Sequent   -- list of known hypothesis
 goal   : Sequent     -- goal to be proved
 justification :- map Id to set of Evidence -- body of evidence
 shape    :- set of MetaType   -- ?
 inference  :- set of Inference   -- how to use this conjecture in other proofs?
             --technology dependant, if not tool dependant?
 uses   :- set of Clue    -- conj. applicability
  inv conj ==
   conj.shape <> {}      -- conjectures *must* have a shape
  and
   -- justifications must have at least some evidence
   forall i in set dom conj.justification &
    conj.justification(i) <> {};
```

```
-- Origin = <TYPE> | <FEASIBILITY> | <REFINEMENT_APPL> | <REF_FEASIBILITY> | <INIT> | ... ???

 -- Contexualised predicate: what can be (implicitly) inferred? what's its relevance?
 Sequent ::
  pred : Pred
  ctx  : Context;

 -- Underlying information per sequent
 Context ::
  infered : seq of Sequent -- what can be inferred by the pred in ctx
  step : nat    -- proof step number / depth
  rank : nat    -- some ordering notion (e.g., lexicograph)
  relevance: nat    -- user/AI defined notion of ranking?

  -- TODO: use google-style page ranking? map-reduce style? etc.
  -- TODO: Cliff suggest to use "graph-like" structure, rather than inferred above
  ;

 -- what kind of conjecture is this?
 -- or should it be "Structure"?
 -- TODO: still undecided about this one.
 MetaType =
    <EXPLORATORY>  -- user is playing around
  | <TOY>    -- toy-problem / abstraction
  | <LEMMA>   -- known subpart of a problem
  | <THEOREM>   -- main/top-level goal of interest
  ;

 -- how is this conjecture to be used by the prover/solver?
 -- or are these just part of the "Why" type itself?
 Inference =
    <FORWARD>
  | <BACKWARD>
  | <REWRITE>
  | <TYPE_JUDGEMENT>
  ;

/*
 * tagging inference from Disproof? Extension of the model?
  - history : p-where   = where was it used and succeeded? theory; goal; proof script; etc
       np-where   = where was it used and failed?
       terms    = set of term structure (in case of formulae variation?)
       proofs   = set of proof scripts attempts
  - weights : hr-based results using various other tags?
*/

-------------------------------------------------------------------------------
--- Evidence
---  * more than just proof script (Attempt): record of proof sccafoldings
---  * both positive and negative; passive and active
---  * scores as a function of the user / system
--
--  * Always start with Proof. Go to Disproof if you dare! (lower chess score risk)
--
--  * SCENARIOS:
--   1) proof Attempt : belief conjecture is true, can't manage to finish proof (no evidence)
--   2) disproof Hunch : belief conjecture is false/wrong by intuition (no evidence)
--   3) disproof Insight : disproof Hunch +confirmation - prescribes Clue from Hunch (with evidence)
--   4) disproof as proof: disproof Hunch discarded - evolve Disproof to proof Attempt
--   5) disproof as test : disproof Hunch -confirmation - creates a counter example (with evidence)
---
--- Disproof
---  * is Insight/Hunch on the goal (e.g., belief/faith: maybe be right/wrong/misleading)
---  * is NOT a unfinished proof!! [Paradox over !!P <> P]
--  * ranked by strength (of belief); pros/cons given
```

```
--   [R0] counter-example as "Test"       [strongest rank]
--     = explicit values falsifying a goal
--     = helpful for machine learning (HR / AM)
--
--   [R1] false goal/contr. hyp as "Attempt"
--     = explicitly failed proof?
--     = ex. justifiably false goal or contradictory hyp
--
--   [R2] prescriptive as "Insight"   = implicitly failed proof? dead-end example
--     = high-level knowledge of previous failure
--     = general (e.g., per family of terms / operators)
--     = harder evidence than guess, yet softer evidence than R1/R0
--     = ex. op. dist. only when goal shrinks/expands
--     = ex. cat doesn't commute"; it works at home :-D
--
--   [R3] putative as "Hunch"        [weakest rank]
--     = experience based-dead end
--     = care is needed to avoid misguidance (deterrent: chess score idea by user)
--     = problem/structure dependent, yet generalisable?
--     = can evolve into a prescripted insight (go up ranks)
--     = ex. "bulk promotion" example from ONFI flash hardware [R3]->[R2]
--     = ex. "FPU calculator" example from IEEE stad v1/v2   [R3 success]
--     = ex. "THE application" example from Tokeneer (failure) [R3 failed]
--     = ex. easy one "x <= y --> x < y" [R3 success] arithmetic why
--
--  * notion of negative-why: it's why(not P) rather than not (why P)! don't dist!
--   n-why-tag x seq of Sequent x Goal x Attempt
--
--  * to make it precise: it is a user decision to choose/change ranks
--   = users are to be scored according to the "precision" of their hunchs?
--
--  * Where negative-"Clues" come from?
--   only from highest ranks and/or user prescribed?
--
--  * this is the data to be used for learning for suggestions to the user?
--
-- Proof
--  * is rigorous transformation (even just AXIOM tag) of the goal
--  * it includes unfinished proofs!!!
--  * contains specific evidence: proof script + tool-session-tag (SMT?)
--  * ranked by strength (of belief) as well:
--   [R0] proof "Script" or tool-session tag (TP, SMT, etc)
--     = explicit / complete proof script
--
--   [R1] proof "Attempt"
--     = explit user decision tagged as "unfinished" proof
--     = subgoals as isolated part of Attempt (e.g., proof Zoom)
--
--   [R2] new Conjecture from proof Attempt
--     = lemma prescription given proof subgoal
--
--  * notion of why intent: why-tag x seq of Sequent x Goal x Attempt
--   may contain tag for incompletion / trusted / axiom
--
--  * Where "Clues" come from?
--   as either a proof Zoom or a suggested new conjecture? Or both?
--
--  * Gaps as just a special kind of attempt?
--
--  * again, to make it precise: it is a user decision or ranked score? Chess player style score
--
-- SIDE: add a role of witnesses within scripts - like in RODIN explicit witness request for quant?

 -- positive or negative evidence --- all attempt based
```

```
Evidence = Proof | Disproof;

   -- (rigorous) positive evidence:
Proof = Attempt
 --inv prf == (prf.why.pos and prf.worth >= 0 and
 --  Evidence
 --| Trust
 --| Sketch
 | Gap
 | Test
 -- proofs have positive Why (as Why(P)) and cannot be accepted "on trust" - these are Disproofs?
 -- if the user believes it to be true, but can't finish, use <SORRY> instead.
 -- I don't want to differentiate between a unfinished proof (<SORRY> and "trusted" proof from somewhere)
 inv prf == (prf.why.pos and prf.worth.tag <> <TRUSTED>);

-- (loose) negative evidence: all are particular forms of proof attempts
Disproof =
   Test  -- active  (counter example)
 | Insight -- passive (prescriptive solution; cat doesn't commute)
 | Hunch  -- passive (solution on experience)
 --| Attempt -- incomplete disproof characterisation??
 inv dprf == (not dprf.why.pos and dprf.worth.tag <> <SORRY>);



-- still evolving set of tags... = ways to differentiate between attempts
-- trouble between duplication x different notions (they are not DUALS, necessarily)
EvidenceTag =
   <AXIOM> -- axiomatic   (for Proof only)
 | <TRUSTED> -- user insight   (for Disproof only)
 | <SORRY> -- unfinished proof  (for Proof only)
 | <QED>  -- finished proof  (Disproof = Test/Contradiction; Proof = finished goal)
 ;

--?
AttemptKind =
   <BLIND> -- exploratory/don't care: auto, prove-by-reduce, grind, etc
 | <GUIDED> -- grounded WhyM info/evidence for given term
 | <CLUES> -- putative WhyM info/clues    for given term
 | <BECAUSE> -- negative WhyM info/?????    for given term
 ;

-- evidence data
AttemptData ::
 rank: int   -- rank as described above (with Rank, may not need EvidenceTag)
 score: Score  -- learned/inferred score
 tag: EvidenceTag -- what kind of evidence is this?
 ;

-- sequence of tokens - keep it abstract
Script = seq of token;

-- Attempt as intent * Terms involved/highlighted * steps
Attempt ::
 why : Why    -- captured intent
 --hyps: seq of Sequent  -- involved hypothesis [maybe don't need this--- it's in context from Conjecture?]
 --goal: Sequent   -- current goal   [but, could be transformed from the conjecture...]
 tool: Tool    -- source tool (SMT/TP/Oracle)
 kind: AttemptKind  -- what kind of attempt is this?
 worth: AttemptData  -- what is this attempt worth?
 script: Script   -- hard evidence?
 ;
 -- operation: length(script) = number proof steps?

/*
```

```
 Evidence = Attempt
  inv evd ==
   if evd.why.pos then
    evd.worth.rank >= 0
   else
    evd.worth.rank < 0;

 Trusted = Evidence
  inv trst ==
   if trst.why.pos then
    evd.worth.rank


Evidence / Trust / Sketch
*/

 -- An insight is a special attempt with negative Why as "not Why(P)"
 Insight = Attempt
  -- insights are TRUSTED negative Whys
  inv att == --isAttemptInsight(ins); -- allows both positive and negative
     att.worth.tag = <TRUSTED> => not att.why.pos;

 -- A hunch is a special attempt with negative Why and lower rank (higher value).
 Hunch = Attempt
  -- hunches are insights with lower rank (e.g., <= in case of other hunches as insights)
  inv hun == isAttemptInsight(hun) and
       (forall i in set { att | att:Attempt & isAttemptInsight(att) } & i.worth.rank <= hun.worth.rank);

 Test = Attempt
   -- tests must be definite, hence QED
  inv tst == tst.worth.tag = <QED> and
   if tst.why.pos then
    -- as positive evidence, it has weakest rank among Attempts
    (forall i in set { att | att: Attempt } & i.worth.rank <= tst.worth.rank)
   else
    -- as negative evidence, it has strongest rank among Attempts
    (forall i in set { att | att: Attempt } & tst.worth.rank <= i.worth.rank)
  ;


 -- bridging gaps in proofs?
 -- * way of introducing controlled uncertainty?
 --Gap ::
 -- origin : seq of Sequent  -- associated sequents
 -- infunk : seq of Sequent  -- inferred/given unknowns
 -- sgoal : Sequent    -- source goal to bridge
 -- tgoal : Sequent    -- target goal to reach
 -- link : Conjecture   -- where is this gap for??
 -- ;
 -- TODO: a gap is a kind of unfinished proof? or a kind of new conjecture?
 --   I guess we don't need it and it will appear within Proof/Disproof.

 -- a gap is a "jump" within a proof attempt ?
 Gap = Attempt
  inv gp == (gp.why.pos and gp.worth.tag = <SORRY>);

 -- score as calculated relevance either as a user defined function or system-infered suggestion on:
 -- * number of times appearing in rewrites
 -- * % of terms changing x times applied
 --  * AM power-curve of strategies/heuristics
 --  * HR concept scoring
 --  * dead-end from logs
 Score ::
  fixed: int  --
  domain: int
```

```
  learned: int
  inv sc == is not yet specified;

-------------------------------------------------------------------------------
--- Why
--  * ontology of proof intent with scores

 -- corroborative data about Why tags?
 Hints = seq1 of Why;

 -- tentative aspect of Attempt
 Clue :: intent  : Why
   rationale : set of (Evidence | Test)
   score  : Score;

 -- (weighted) ontological tagging
 Why :: kind: WhyTag -- justification category
   score: Score -- relevance/importance
   pos: bool  -- positive / negative intent
   ;

 -- should we have separate or composite Why categories?
 WhyTag =  --Operators
   <OP_ORDER> | <OP_COMB> | <DIST_OP> | <DIFF_OP> | <COMM_OP> |
     -- Witnesses / quantifiers
   <FORALL_NEEDED> | <EXISTS_NEEDED> |
     -- Strategies
   <INDUCTION> | <NF_REDUCTION> | <LR_REWRITE> |
   -- Transformations
   <DATATYPE_MAPPING> | <EQ_TRANSF> |
   -- Decision procedures
   <ARITHMETIC> | <SMT> | <QUICKCHECK> |
   -- Eureka introduction: infered? user tags?
   <EUREKA> | <THOR_HAMMER> |
   -- Log / plans / AI ?
   <INF_FROM_LOG> | <HR_CLUE> |
   -- Negative? Should it be a diff tag? NWhyTag?
   <NOT_THEOREM>;

-------------------------------------------------------------------------------
--- Auxiliary term structure
---
---  ? try not to depend on any expr/pred term structure, but rather what is
---    needed abstractly.
--  ? maybe necessary if learning is about structure; or not if ontological.

/*
 --------------------
 -- Expr tree (explicit)
 Expr = Const | Var | Cond; --| OperatorDefn?

 Const ::  nat | bool;
 Var  :: id: Id
    tp: [<Bool> | <Nat>];
 Cond :: test: Expr
    yes : Expr
    no : Expr
 inv mk_Cond(b, y, n) ==
  ((is_Const(b) and is_bool(b)) or
   (is_Var(b) and b.tp = <Bool> ))--is_Bool(b.tp))
  and
  (is_Expr(y) and is_Expr(n));

 --------------------
 -- Pred tree (explicit)
```

```
 Pred = Atom | And | Not | ForAll;

 Atom = bool | Eq;

 Eq  ::  lhs : Expr
    rhs : Expr;

 And  ::  lhs : Pred
    rhs : Pred;

 Not  :: p : Pred;

 ForAll :: vars: seq of Var
    pred: Pred
 inv mk_ForAll(v, -) ==
  --(is_Var(v) and is_Pred(p) and -- do I need those typing things?
  forall i, j in set elems v & i <> j;
*/

 TermData ::
  bv: set of Id -- bounded variables
  fv: set of Id -- free variables
   inv td == td.bv inter td.fv = {} and
       td.bv union td.fv <> {};

 Atom ::
  vars: TermData -- information of interest?
  body: token; -- abstract structure

 -----------------------
 -- Expr tree (implicit)
 Expr ::
  exp: Atom  -- expression structure
  typ: Signature -- type information
  val: Value;  -- current bindings?

 -----------------------
 -- Pred tree (implicit)
 PredTerm ::
  pred: Atom  -- predicate structure
  val: bool;  -- current bindings?

 EqTerm ::
  lhs: Expr
  rhs: Expr; -- equality is special?

 Pred = PredTerm | EqTerm;

 Term = Expr | Pred;

 -- TODO: cross-fertilise these data structure with Isabelle / Mural term structure?

 -----------------------
 -- Underspecified terms

 -- Id's are used to "tag" various structures.
 Id = token;

 -- Follows from TP/tool type hierarchy?
 Signature = token;

 -- Follows from TP/tool proof scripts?
 --Attempt  = token;

 -- ??? string to the executable ???
```

```
 Tool  = token;

 --- log-data (PSP-inspired)
 Log   = token;

 -- binding values to variables / expr; depends on type system
 Value  = token

 -- TODO:CHECK: what's better style, named types for everything
 --       (e.g., IdSet) or direct use (e.g., set of Id)?
 --
 --IdSet   = set of Id;
 --OperatorMap  = map Operator to OpDefn;
 --ConjectureMap = map Id to Conjecture;
 --

-- TODO: how to make such constant sets?
--values
-- allInsights: set of Attempt = { att | att:Attempt & is_Insight(att) };

functions

-- TODO: do I need this extra layer of type checking for attempt given as is_XXX(att)? Not?

 -- disproof attempts are trusted and with negative Why (as "not Why(P)")
 --isValidInsight: Attempt -> bool
 --isValidInsight(att) == (/*is_Insight(att) and*/ att.worth.tag = <TRUSTED> and not att.why.pos);

 -- a test is definite, hence QED as the only option. it might be either positive or negative
 --isValidTest: Attempt -> bool
 --isValidTest(att) == (/*is_Test(att) and*/ att.worth.tag = <QEQ>);

 -- a valid proof must be positive evidence and cannot be given on trust
 --isValidProof: Attempt -> bool
 --isValidProof(att) == (is_Proof(att) and att.worth.tag <> <TRUSTED> and att.why.pos);

 -- if a given attempt is a proof, then it is finished if QED tagged
 --finishedProof: Attempt -> bool
 --finishedProof(att) == (isValidProof(att) => att.worth.tag = <QED>);

 --   *USE => in FINISHED PROOF TO MAKE (not finishedProof) unfinished proof*
 --unfinishedProof: Attempt -> bool
 --unfinishedProof(att) == (isValidProof(att) and att.worth.tag <> <QED>);

 -- attempts taken on trust can only be negative evidence
 -- conversely, if not taken on trust, then it is positive evidence
 isAttemptInsight: Attempt -> bool
 isAttemptInsight(att) == att.worth.tag = <TRUSTED> => not att.why.pos;

 linked: Attempt * Sigma -> bool
 linked(a, s) == is not yet specified;

 complete: Attempt * Sigma -> bool
 complete(a, s) == is not yet specified;

 analyse:  Conjecture -> set of Why
 analyse(c) == is not yet specified;

 evolve: Disproof -> Proof
 evolve(d) == is not yet specified;

/*
 rankHyp : Conjecture * Ordering -> Conjecture
 -- order the hypothesis in some way
*/
```

```
/*
 * generate POs
 * start [new] proof
 * got stuck: review / inform / judge
  - proof critics? others
 * proof plan execution
  - splitting / induction / rewriting
 * proof engineering needed
 * identification
  - missing hypothesis
  - ranking hypothesis
  - disproofs (via intuition)
  - counter examples (explicit)
 * Eureka introduction
 * Thor's hammer (force throuhg; isar Sorry)
 * classify PO / goal
  - attempts: blind   = auto, p-by-reduce, grind, etc
     with-intent(X) = grounded WhyM info for X
     using-clues(Y) = putative WhyM info for Y
     becauseof(Z) = negative WhyM info for Z
  - metatype: exploration  = user is playing around
     toy [abstration]= toy-problem construction (!!)
     lemma [subpart] = known subpart
 * create [choose] strategy
  - split, induct, rewrite, generalise
 * tagging terms / lemmas (user burden: rationale = better learning?)
  - op/defn : isFcn/Op, Const, Dest, etc
  - prop-sets : p-assoc(op1, op2), p-commu, p-dist-l/r,
  - anti-props: np-assoc(op1, op2), etc..
  - conceptual: core     = considered basic by the user
       extension   = refinement of "core"
       invariant   = general property predicate/term
  - structural: pointwise   = explicit consider every point of set or function
       set-based   = set-theoretical proof/structure (don't expand to elements)
       extension   = equality as quantified comparison (x=y iff (ALL i: x @ i : y) etc)
       liebinitz   = equality as substitution over equals-for-equals
  - inference : rule    = to be used a rewrite rule? forw/backw?
       tjudgement  = to be used as type judgements?
  - origin : extensional = from the problem text itself
       intentional = from user insight
  - intent : desc    = textual description of how this is to be used?
       related   = link to related terms / lemmas / defs (possible to infer?)
  - witnesses : concrete examples for quantifiers / predicates?
 * tagging inference
  - history : p-where   = where was it used and succeeded? theory; goal; proof script; etc
       np-where   = where was it used and failed?
       terms    = set of term structure (in case of formulae variation?)
       proofs   = set of proof scripts attempts
  - weights : hr-based results using various other tags?
 * disproofs
  - passive (insight), active (counter-example), gaps, etc
 * ontology of lemmas / terms
  - Paolo's provenance stuff?
  - WhyM scoring function?
  - success / failure applicability rate?
 * classification of lemmas / terms
  - postmortem analysis: weight/score adjustments
  - suggestion of lemmas; patches; fixes; strategies; etc
 * propose gap
  - justified / tentative lemmas?
  - infered from theory usage / clues?
 * proof refactoring / reconstruction / clean-up / synthesis
  - look for Iain Whitehouse(?) Edinburgh work on Isar transf.
  - look for MSR guy from UV10 talking about synthesis techniques
```

```
 * armageddon introduction
  - badly stuck / lost; worth resetting the world :-) !!
 * metamorphosis introduction
  - isomorphic lemma / proof suggestion
  - equivalence lemmas for proof engineering (e.g., eq def easier to prove)
  - change / refine underlying datatype

 * LEARNING (!!!)
  - HR concept formation and relevance-measurement using WhyM data (e.g., Disproof/Gap)
  - AM power-curves and concept formation ideas?
  - Improved notions for scoring (see TP term indexing, strategies, SMT stuff, etc)

 # DNA analogy: 2% gene encoding; 98% "garbage DNA"->"DNA mechanics"
     (proof scripts); (proof sessions)->(proof intent/why)




 # TOOL-ARCHITECTURE:
  - WhyM in VDM (Overture) enables:
   + formal documentation of proof intent (meta-proving specification)
   + trace / animation analysis options
   + POG for intra consistency (WhyM is sound / feasible)
   + POG for exo consistency (WhyM suggesting extra POs PScripts)
   + closer to FM tools for actual proof (TP / SMT / etc)
   + JML / Java code generation of WhyM
   + formal link with PSP-inspired logging?

  - Ecore/EMF/Epsilon world
   + loose manipulation / transformation of WhyM instances
   + visual appeal / user interfacing
   + shared effort on MDA (model driven architectures)
   + meta-modelling support (EVL, EWL, EOL, ECL, etc).
   + AI transformations
   + Method agnostic (not in VDM, say).

*/

end WhyM
```

Function or operation	Coverage	Calls
WhyM	0.0%	0
analyse	0.0%	0
complete	0.0%	0
evolve	0.0%	0
isAttemptInsight	0.0%	0
linked	0.0%	0
WhyM.vdmsl	0.0%	0

3 WhyM0

```
module WhyM0

definitions
```

```
types

------------------------------------------------------------------------------
--- Cliff's "world"
---  * AI4FM strategy language

 Sigma :: map Id to Theory;  -- state for models of why operations

------------------------------------------------------------------------------
--- Theory
---  * Theory records intent information and inference

 Theory :: specialises : Id       -- dependencies down
    log   : Log       -- associated log
    uses  : set of Id      -- dependencies up
    ttypes  : set of Id      -- known types
    operators : map Operator to OpDefn  -- tagged/known ops
    results  : map Id to Conjecture   -- proved conjectures
    strategies : set of (Why * (seq of Why))  -- ??
 --inv  TODO

 -- TODO: also comparative/ordering/ranking notions
 --   between various Theory components / structs
 ;

------------------------------------------------------------------------------
--- Operators
---  * Augumented from tool term structure with meta-tagging?

 Operator :: id  : Id
    cat  : OpCategory
    assoc : OpAssoc
    prec : nat;

 OpCategory  = <Relation> | <Function>; --| <Generic>;
 OpAssoc  = <ASSOC_LEFT> | <ASSOC_RIGHT>; --| <ASSOC_NONE>;

 OpDefn :: type: Signature
    defn: Definition;

------------------------------------------------------------------------------
--- Conjectures
---  * Augumented from tool term structure with meta-tagging + Evidence

 -- conj as in seq. calculus plus extra model Why info.
 -- assuming only "goal" identify a conjecture (used ":-")
 Conjecture :: hypothesis  :- seq of Sequent
     goal   : Sequent

     -- judgement (as field name) = overloaded term (used by TP)
     --
     -- have evidence as sets: Alan's n-proofs per conjecture?
     -- plus more than one possible disproof as well?
     justification :- map Id to set of Evidence
     shape    :- set of MetaType
     uses   :- set of Clue
  inv conj ==
    -- all conjectures *must* have a shape
    conj.shape <> {}
   and
    -- all justifications must have some evidence
    forall i in set dom conj.justification &
     conj.justification(i) <> {};

 -- Contexualised predicate: what can be (implicitly) inferred? what's its relevance?
```

```
Sequent ::  pred : Pred
    ctx  : Context;

-- Underlying information per sequent
Context ::  infered : seq of Sequent -- what can be inferred by the pred in ctx
    step : nat     -- proof step number / depth
    rank : nat     -- some ordering notion (e.g., lexicograph)
    relevance: nat     -- user/AI defined notion of ranking?

 -- TODO: use google-style page ranking? map-reduce style? etc.
 ;

--------------------------------------------------------------------------------
--- Evidence
---  * more than just proof script (Attempt): record of proof sccafoldings
---  * both positive and negative; passive and active
---  * scores as a user/system function

Evidence  = Proof | Disproof | Gap;

EvidenceTag = <AXIOM> |  -- axiomatic
     <TRUSTED> | -- user insight
    Tool |   -- tool baased (SMT/TP/Oracle)
    Attempt  -- script based (Isar)
    ;

-- (rigorous) positive evidence documentation
Proof   =  EvidenceTag ;

-- (loose) negative evidence: may be just user insight
Disproof = Insight | -- passive (user insight)
   Test  -- active  (counter example)
   ;

-- Route * (Terms involved/highlighted) * Depth/Step no.
Insight = EvidenceTag * set of Sequent * nat
 -- cannot use axioms
 inv ins ==
  (ins.#1 <> <AXIOM>);

-- Counter example from goal
--  * what is a test predicate here? A counter example? Should it be (seq of Pred)?
  --pred : (seq of Expr) * Expr * (set of Conjecture) -> bool [Cliff's original]
-- * score as a function of its use/importance?
--
Test :: value : Expr     -- actual counter example
  origin : seq of Sequent  -- associated sequents
  score : Score;    -- calculated relevance

-- score as a user defined function? or system-infered suggestion on
-- * number of times appearing in rewrites
-- * % of terms changing x times applied
--  * AM power-curve of strategies/heuristics
--  * HR concept scoring
--  * dead-end from logs
Score = int
 inv sc == is not yet specified;

-- bridging gaps in proofs?
-- * way of introducing controlled uncertainty?
Gap :: origin : seq of Sequent  -- associated sequents
  infunk : seq of Sequent  -- inferred/given unknowns
  sgoal : Sequent    -- source goal to bridge
  tgoal : Sequent    -- target goal to reach
  link : Conjecture   -- where is this gap for??
```

```
   ;

------------------------------------------------------------------------------
--- Why
--  * ontology of proof intent with scores

 -- tentative aspect of Attempt
 Clue :: intent  : Why
   rationale : set of (Evidence | Test)
   score  : Score;

 -- (weighted) ontological tagging
 Why :: kind: WhyTag -- justification category
   score: Score -- relevance/importance
   ;

 -- should we have separate or composite Why categories?
 WhyTag =  --Operators
   <OP_ORDER> | <OP_COMB> | <DIST_OP> | <DIFF_OP> | <COMM_OP> |
     -- Witnesses / quantifiers
   <FORALL_NEEDED> | <EXISTS_NEEDED> |
     -- Strategies
   <INDUCTION> | <NF_REDUCTION> | <LR_REWRITE> |
   -- Transformations
   <DATATYPE_MAPPING> | <EQ_TRANSF> |
   -- Decision procedures
   <ARITHMETIC> | <SMT> | <QUICKCHECK> |
   -- Eureka introduction: infered? user tags?
   <EUREKA> | <THOR_HAMMER> |
   -- Log / plans / AI ?
   <INF_FROM_LOG> | <HR_CLUE>;

------------------------------------------------------------------------------
--- Auxiliary term structure (for playing around)

 --------------------
 -- Expr tree
 Expr = Const | Var | Cond; --| OperatorDefn?

 Const ::  nat | bool;
 Var  :: id: Id
    tp: [<Bool> | <Nat>];
 Cond :: test: Expr
    yes : Expr
    no : Expr
 inv mk_Cond(b, y, n) ==
  ((is_Const(b) and is_bool(b)) or
   (is_Var(b) and b.tp = <Bool> /*is_Bool(b.tp)*/))
  and
  (is_Expr(y) and is_Expr(n));

 --------------------
 -- Pred tree
 Pred = Atom | And | Not | ForAll;

 Atom = bool | Eq;

 Eq  ::  lhs : Expr
    rhs : Expr;

 And  ::  lhs : Pred
    rhs : Pred;

 Not  :: p : Pred;
```

```
 ForAll :: vars: seq of Var
    pred: Pred
 inv mk_ForAll(v, -) ==
  --(is_Var(v) and is_Pred(p) and -- do I need those typing things?
  forall i, j in set elems v & i <> j;


 -- TODO: cross-fertilise these data structure with Isabelle / Mural term structure

 ------------------------
 -- Underspecified terms

 -- Id's are used to "tag" various structures.
 Id = token;

 -- Follows from TP/tool type hierarchy?
 Signature = token;

 -- Follows from TP/tool term structure?
 Definition = token;

 -- Follows from TP/tool proof scripts?
 Attempt  = token;

 -- ??? string to the executable ???
 Tool  = token;

 --- log-data (PSP-inspired)
 Log   = token;

 -- ???
 MetaType = token;

 -- TODO:CHECK: what's better style, named types for everything
 --       (e.g., IdSet) or direct use (e.g., set of Id)?
 --
 --IdSet   = set of Id;
 --OperatorMap  = map Operator to OpDefn;
 --ConjectureMap = map Id to Conjecture;
 --

functions

 linked: Attempt * Sigma -> bool
 linked(a, s) == is not yet specified;

 complete: Attempt * Sigma -> bool
 complete(a, s) == is not yet specified;

 analyse:  Conjecture -> set of Why
 analyse(c) == is not yet specified;

/*
 rankHyp : Conjecture * Ordering -> Conjecture
 -- order the hypothesis in some way
*/

/*
 * generate POs
 * start [new] proof
 * got stuck: review / inform / judge
  - proof critics? others
 * proof plan execution
  - splitting / induction / rewriting
 * proof engineering needed
```

```
* identification
 - missing hypothesis
 - ranking hypothesis
 - disproofs (via intuition)
 - counter examples (explicit)
* Eureka introduction
* Thor's hammer (force throuhg; isar Sorry)
* classify PO / goal
 - attempts: blind   = auto, p-by-reduce, grind, etc
    with-intent(X) = grounded WhyM info for X
    using-clues(Y) = putative WhyM info for Y
    becauseof(Z) = negative WhyM info for Z
 - metatype: exploration  = user is playing around
    toy [abstration]= toy-problem construction (!!)
    lemma [subpart] = known subpart
* create [choose] strategy
 - split, induct, rewrite, generalise
* tagging terms / lemmas (user burden: rationale = better learning?)
 - op/defn : isFcn/Op, Const, Dest, etc
 - prop-sets : p-assoc(op1, op2), p-commu, p-dist-l/r,
 - anti-props: np-assoc(op1, op2), etc..
 - conceptual: core     = considered basic by the user
      extension   = refinement of "core"
      invariant   = general property predicate/term
 - structural: pointwise   = explicit consider every point of set or function
      set-based   = set-theoretical proof/structure (don't expand to elements)
      extension   = equality as quantified comparison (x=y iff (ALL i: x @ i : y) etc)
      liebinitz   = equality as substitution over equals-for-equals
 - inference : rule    = to be used a rewrite rule? forw/backw?
      tjudgement  = to be used as type judgements?
 - origin : extensional = from the problem text itself
      intentional = from user insight
 - intent : desc    = textual description of how this is to be used?
      related   = link to related terms / lemmas / defs (possible to infer?)
 - witnesses : concrete examples for quantifiers / predicates?
* tagging inference
 - history : p-where   = where was it used and succeeded? theory; goal; proof script; etc
      np-where   = where was it used and failed?
      terms    = set of term structure (in case of formulae variation?)
      proofs   = set of proof scripts attempts
 - weights : hr-based results using various other tags?
* disproofs
 - passive (insight), active (counter-example), gaps, etc
* ontology of lemmas / terms
 - Paolo's provenance stuff?
 - WhyM scoring function?
 - success / failure applicability rate?
* classification of lemmas / terms
 - postmortem analysis: weight/score adjustments
 - suggestion of lemmas; patches; fixes; strategies; etc
* propose gap
 - justified / tentative lemmas?
 - infered from theory usage / clues?
* proof refactoring / reconstruction / clean-up / synthesis
 - look for Iain Whitehouse(?) Edinburgh work on Isar transf.
 - look for MSR guy from UV10 talking about synthesis techniques
* armageddon introduction
 - badly stuck / lost; worth resetting the world :-) !!
* metamorphosis introduction
 - isomorphic lemma / proof suggestion
 - equivalence lemmas for proof engineering (e.g., eq def easier to prove)
 - change / refine underlying datatype

* LEARNING (!!!)
 - HR concept formation and relevance-measurement using WhyM data (e.g., Disproof/Gap)
```

```
  - AM power-curves and concept formation ideas?
  - Improved notions for scoring (see TP term indexing, strategies, SMT stuff, etc)

 # DNA analogy: 2% gene encoding; 98% "garbage DNA"->"DNA mechanics"
     (proof scripts); (proof sessions)->(proof intent/why)




 # TOOL-ARCHITECTURE:
  - WhyM in VDM (Overture) enables:
   + formal documentation of proof intent (meta-proving specification)
   + trace / animation analysis options
   + POG for intra consistency (WhyM is sound / feasible)
   + POG for exo consistency (WhyM suggesting extra POs PScripts)
   + closer to FM tools for actual proof (TP / SMT / etc)
   + JML / Java code generation of WhyM
   + formal link with PSP-inspired logging?

  - Ecore/EMF/Epsilon world
   + loose manipulation / transformation of WhyM instances
   + visual appeal / user interfacing
   + shared effort on MDA (model driven architectures)
   + meta-modelling support (EVL, EWL, EOL, ECL, etc).
   + AI transformations
   + Method agnostic (not in VDM, say).

*/


end WhyM0
```

Function or operation	Coverage	Calls
WhyM0	0.0%	0
analyse	0.0%	0
complete	0.0%	0
linked	0.0%	0
WhyM0.vdmsl	0.0%	0

4 WhyMCliff

```
module WhyMCliff

definitions

types
 Sigma :: map Id to Theory;

 Theory :: specialises : Id
    uses  : set of Id
    ttypes  : set of Id
    operators : map Operator to OpDefn
    results  : map Id to Conjecture
    strategies : set of (Why * (seq of Why));

 OpDefn :: type: Signature
```

```
    defn: Definition;

 Conjecture :: hypothesis  : seq of Judgement
     goal    : Judgement
     justification :- map Id to Proof | Disproof -- should these be ":-" or juts ":"?
     shape     :- set of MetaType
     uses    :- set of Clue;

 Proof = <AXIOM> | <TRUSTED> | Tool | Attempt;
 Disproof = <TRUSTED> | Tool | Attempt;

 Clue ::  intent : Why
    evidence: Evidence;

 Why =  <OP_ORDER> | <OP_COMB> | <DIST_OP> | <DIFF_OP> | <COMMUTE_OP> |
     <FORALL_NEEDED> | <EXISTS_NEEDED> |
     <INDUCTION> | <NF_REDUCTION> | <LR_REWRITE> |
   <DATATYPE_MAPPING> | <EQ_TRANSF>;

 -- Id's are used to "tag" various structures.
 Id = token;

 -- Entities that are to be kept abstract up to this stage
 Operator  = token;
 Signature = token;
 Definition = token;
 Judgement = token;
 Evidence = token;

 Tool  = token;
 Attempt  = token;
 MetaType = token;

functions

 linked: Attempt * Sigma -> bool
 linked(a, s) == is not yet specified;

 linkedImplicit(a: Attempt, s: Sigma) r: bool
 pre true
 post true;

 complete: Attempt * Sigma -> bool
 complete(a, s) == is not yet specified;

 analyse:  Conjecture -> set of Why
 analyse(c) == is not yet specified;

end WhyMCliff
```

Function or operation	Coverage	Calls
WhyMCliff	0.0%	0
analyse	0.0%	0
complete	0.0%	0
linked	0.0%	0
linkedImplicit	0.0%	0
WhyMCliff.vdmsl	0.0%	0