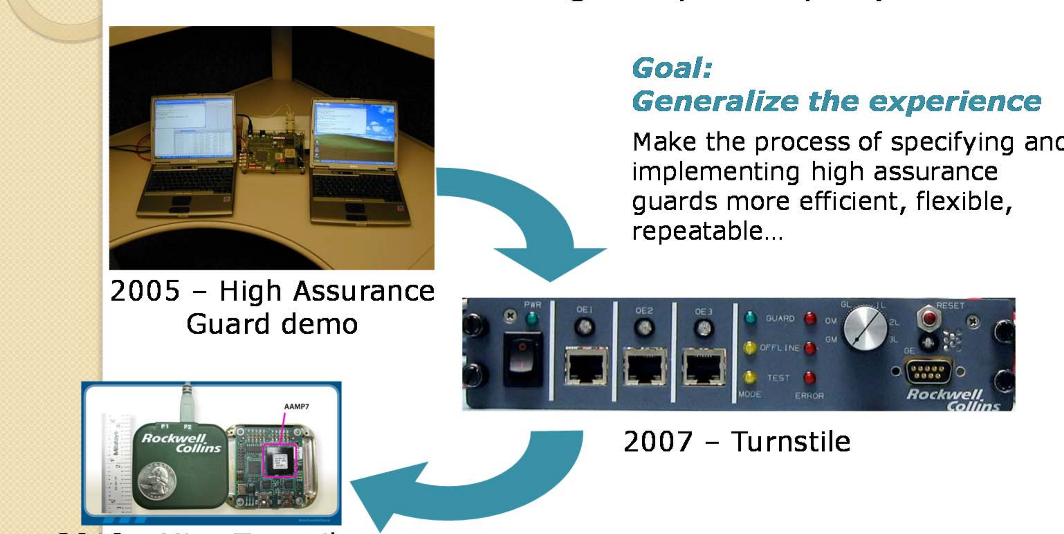
The Guardol Language and Verification System

Hardin, Slind, Whalen, and Pham

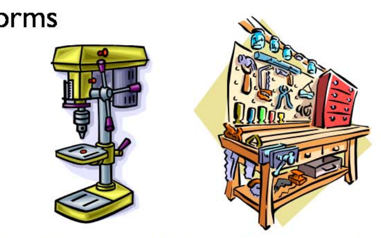
Guardol Concept

A *guard* mediates information sharing between security domains according to a specified policy



Guardol Objectives

- Develop Domain Specific Language for guards
  - Specification language dedicated to a particular problem domain, representation style, solution technique
- Automate the design flow
  - Analysis and implementation artifacts automatically generated with high assurance
- Integrated analysis capabilities
  - Model checking of key requirements from guard specification
  - Information flow analysis to verify correct data paths and separation
- Support for a wide variety of guard platforms
  - Configurable guards
  - Custom hardware, embedded software
  - Open/standard platforms



Typical Guard Operations

- packet observations – reading field values of the packet
- packet dropping – removal of an entire packet from the stream
- packet transformation – changing the value of fields in a packet
- packet expansion – adding new fields to a packet
- packet contraction – removing fields from a packet
- packet generation – construction of audit messages

Example: Dirty Word Search

```
type Msg = string;
type Result = { Ok: Msg | Audit: string };
imported function DIRTY_WORD_SEARCH(Text : in Msg, Output : out Result);
```

Examples

```
DIRTY_WORD_SEARCH("This is CLASSIFIED data", Output);
   Output: Ok => "This is ---------- data"

DIRTY_WORD_SEARCH("This is SECRET data", Output);
   Output: Audit => "Secret data detected. Message deleted."
```


Dirty Word Search on Trees

```
type MsgTree = { Leaf | Node: [Value: Msg; Left: MsgTree; Right: MsgTree] };
type ResultTree = { Ok: MsgTree | Audit: string };

function DWS_TREE(Input : in MsgTree, Output : out ResultTree) = begin
var
  ValueResult : Result;
  LeftResult, RightResult : ResultTree;
in
  match Input with begin
    MsgTree'Leaf => Output := ResultTree'Ok(MsgTree'Leaf);
    MsgTree'Node node => begin
      DIRTY_WORD_SEARCH(node.Value, ValueResult);
      match ValueResult with begin
        Result'Audit A => Output := ResultTree'Audit(A);
        Result'Ok ValueMsg => begin
          ...
            Output := ResultTree'Ok(MsgTree'Node
               [ Value: ValueMsg, Left: LeftMsgTree, Right: RightMsgTree ]);
```

Guardol = Ada + ML-Style Datatypes

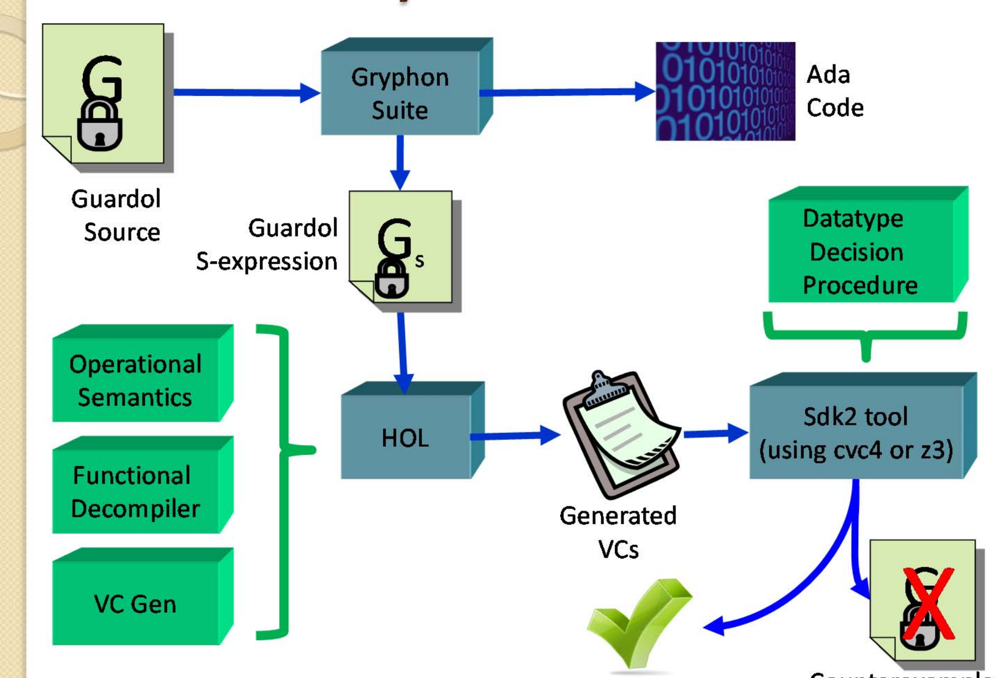
Desirable property: if message is emitted, then it contains no dirty words

Property Specification

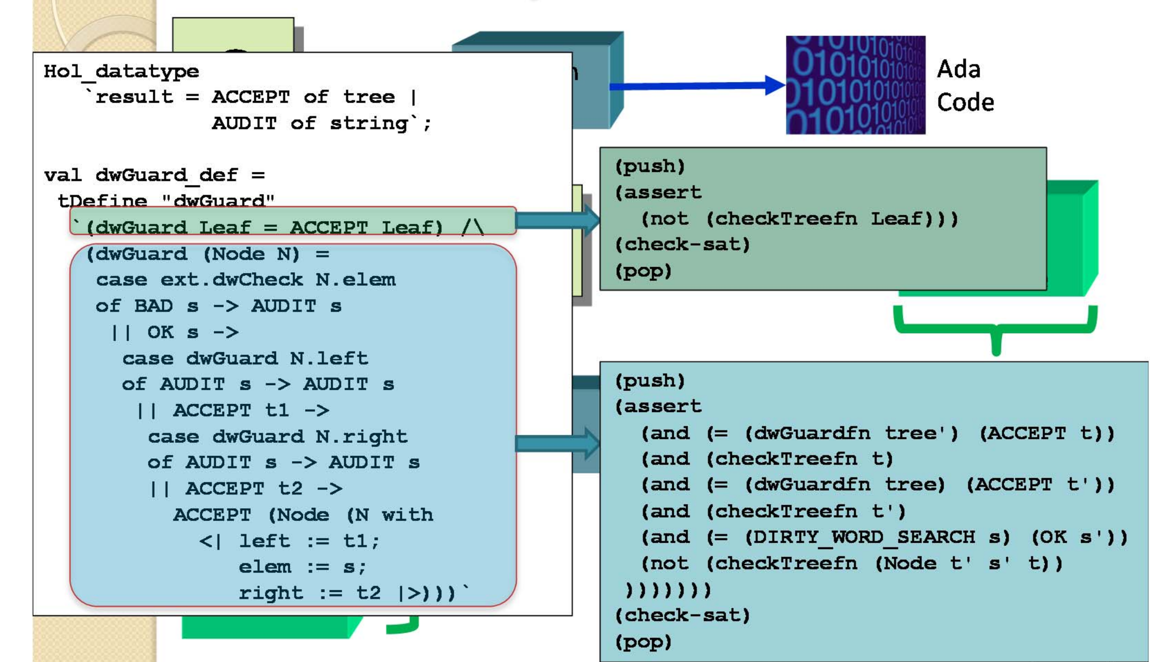
```
spec DWS_TREE_Correct = begin
var
  MT : MsgTree;
  RT : ResultTree;
in
  if (forall (M:Msg).DWS_Idempotent(M))
  then begin
    DWS_TREE(MT, RT);
    match RT with begin
      ResultTree'Ok MT2 =>
         check DWS_TREE_Check(MT2);
      ResultTree'Audit A => skip;
    end
  end
  else
    skip;
end
```

```
function DWS_TREE_Check(MT : in MsgTree)
   returns Output : bool = begin
var
  R : Result;
in
  match MT with begin
    MsgTree'Leaf => Output:=true ;
    MsgTree'Node node => begin
      DIRTY_WORD_SEARCH(node.Value, R);
      match R with begin
        Result'Ok M =>
          Output:=(node.Value = M);
        Result'Audit A =>
          Output:=false;
      end
      Output := Output and
       DWS_TREE_Check(node.Left) and
       DWS_TREE_Check(node.Right);
    end
  end
end
```


Guardol Analysis Architecture



Guardol Analysis Architecture



Verification Conditions

```
(assert
  (and (= (dwGuardfn tree') (ACCEPT t))
  (and (checkTreefn t)
  (and (= (dwGuardfn tree) (ACCEPT t'))
  (and (checkTreefn t')
  (not (checkTreefn (Node t' s' t)))
  …
where
(define-fun checkTreefn ((MT MsgTree)) bool
  (ite (is_MsgTree_Leaf MT)
   true
   (and
     (let ((R (DIRTY_WORD_SEARCH (MsgTree_value MT))))
      (and (is_Result_Ok R)
           (= (Result_Value R) M)))
     (checkTreefn (MsgTree_left MT))
     (checkTreefn (MsgTree_right MT)))))
```

Quantifier-free Formula

Collection Domain with a Decidable Theory

Generalized Fold Function

Suter-Dotta-Kuncak Procedure

- Purification

```
(and (= (dwGuardfn tree) (ACCEPT t'))
 (and (checkTreefn t')
 (not (checkTreefn (Node t' s' t))) …

   (and (= t1 (Node t' s' t)) (= t2 (ACCEPT t')))
   (and (= c1 (checkTreefn t')) (= c2 (checkTreeFn t1)))
   (and (= (dwGuardfn tree) t2) c1 (not c2))
```

- Unification & Partial Evaluation
  - Structural unification of tree terms
  - …followed by partial evaluation of catamorphism (fold) function over unified trees
- Afterwards: formula in *collections theory*
- Important quality: **completeness**

So, what can we prove?

- Properties of message fields:
  - Field *z* is always > 0
  - Field *z* contains less than 50 characters
  - Field *z* has no dirty words
  - Field *z* has precision no better than 10 meters
  - Applying *f* to *z* leaves *z* unchanged
- Properties of abstractions of message fields:
  - The sum of all integer fields is < 100.
  - The number of dirty words is < 10.
  - The set of all fields in the message contains *y*.
- Relations of abstractions between abstractions of pre- and post-messages.
  - If *x* contained no dirty words prior to processing, then it contains no dirty words after processing.

Conclusion

- Guardol: New DSL and tool suite for reasoning about guards
  - Has nice guarantees for certain kinds of properties
  - Language syntax is reasonable to non-geeks
  - Tools and architecture designed for rigor from specification through to implementation
- Decision procedures [SDK] are now able to prove interesting properties of unbound data & computation
  - We have an open-source tool available [UMN].
- Lots of interesting future work involving
  - Language improvements
  - Integration of string decision procedures
  - Extending SDK completeness results
  - Support for intentional properties