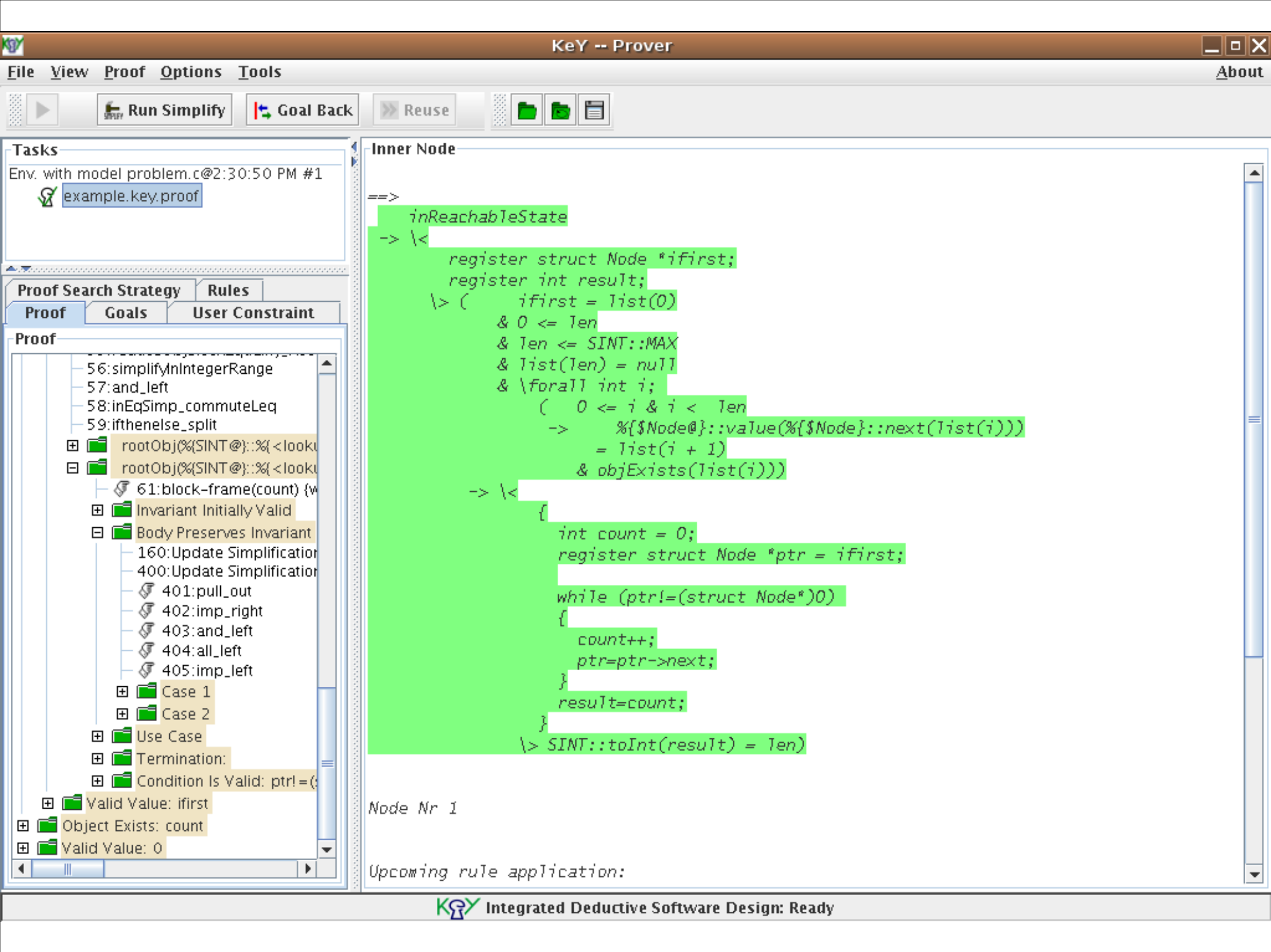
Deductive Verification of C

Oleg Mürk

KeY Project

- Dynamic logic for imperative languages
- Java(Card), C, ...
- Interactive proof assistant
- Usability, Automation, Easy to learn
- Operation contracts & invariants


KeY -- Prover
File View Proof Options Tools
About
Run Simplify
Goal Back
Reuse
Tasks
Env. with model problem.c@2:30:50 PM #1
example.key.proof
Proof Search Strategy
Rules
Proof
Goals
User Constraint
Proof
56:simplifyInIntegerRange
57:and_left
58:inEqSimp_commuteLeq
59:ifthenelse_split
rootObj(%(SINT@):::%(<looku
rootObj(%(SINT@):::%(<looku
61:block-frame(count) {w
Invariant Initially Valid
Body Preserves Invariant
160:Update Simplificatior
400:Update Simplificatior
401:pull_out
402:imp_right
403:and_left
404:all_left
405:imp_left
Case 1
Case 2
Use Case
Termination:
Condition Is Valid: ptr!=(
Valid Value: ifirst
Object Exists: count
Valid Value: 0
Inner Node
==>
inReachableState
-> \<
register struct Node *ifirst;
register int result;
\> (ifirst = list(0)
& 0 <= len
& len <= SINT::MAX
& list(len) = null
& \forall int i;
(0 <= i & i < len
-> %{$Node@}::value(%{$Node}::next(list(i)))
= list(i + 1)
& objExists(list(i)))
-> \<
{
int count = 0;
register struct Node *ptr = ifirst;
while (ptr!=(struct Node*)0)
{
count++;
ptr=ptr->next;
}
result=count;
}
\> SINT::toInt(result) = len)
Node Nr 1
Upcoming rule application:
Integrated Deductive Software Design: Ready

My Work

- C Dynamic Logic (CDL)
  - ANSI C
    - Portable, type-safe
  - Specializations
    - MISRA-C
    - IA-32
    - ...
- Language variability of KeY
- KeY-C

Dynamic Logic

- (Typed) First Order Logic
- Modalities: $\phi \rightarrow \langle P \rangle \psi \,, \quad \phi \rightarrow [P] \psi$
- Rigid vs Non-rigid Symbols
- Kripke Structure
- Updates: $\{ \ \mathrm{value}(obj) := val \ \| \ \mathrm{next} := \mathrm{next} + 1 \ \} \phi$
- Sequent Calculus

CDL: Challenges

- Hierarchical data structures
- Integers have multiple representations
- Objects can be deleted
- Pointers to members, local variables
- (Non-deterministic expressions)
- (Untyped memory access)

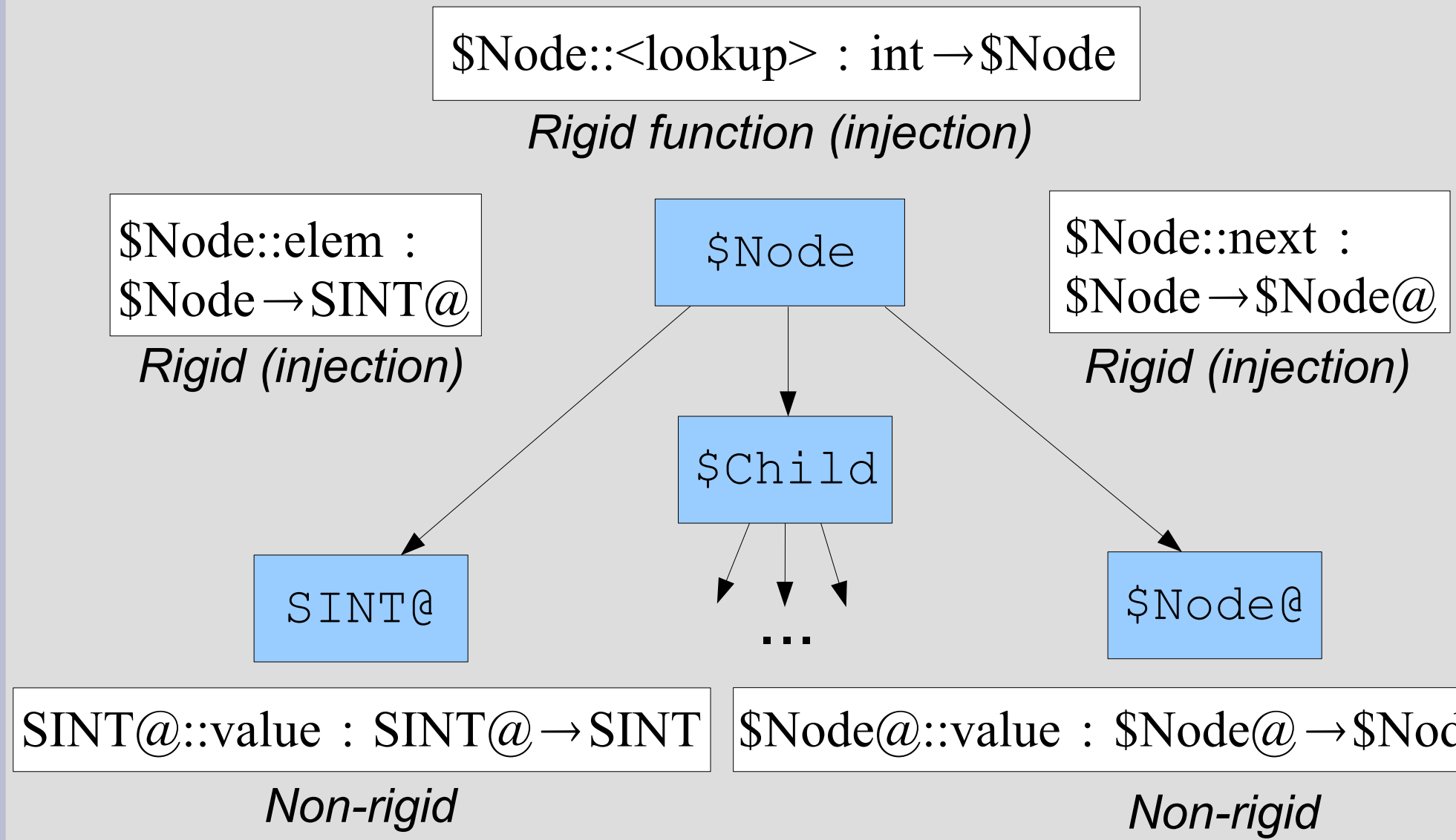
CDL: Challenges

- Vague specification
  - No official type system
  - No official formal semantics
- Lots of **underspecification**
- Some **non-determinism**

Memory Model

```
struct Node {
    int elem;
    struct Child child;
    struct Node* next;
};
```

Type-Safe Heap

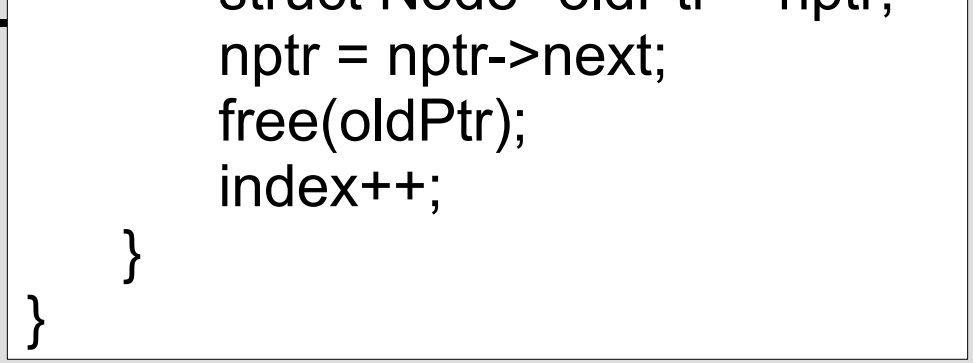


Current Status

- Types
  - Integer, Pointer, Scalar, Struct, Array
- WHILE language with
  - C declarations
  - C expressions
  - Compound statements
  - If, while statements

Example Program

```
int count = 0;
struct Node* ptr = ifirst;
while (ptr != (struct Node*)0) {
      count++;
      ptr = ptr->next;
}
int* arr = (int*)0;
if (count > 0) {
      ...
}
*ocount = count;
*oarr = arr;
```

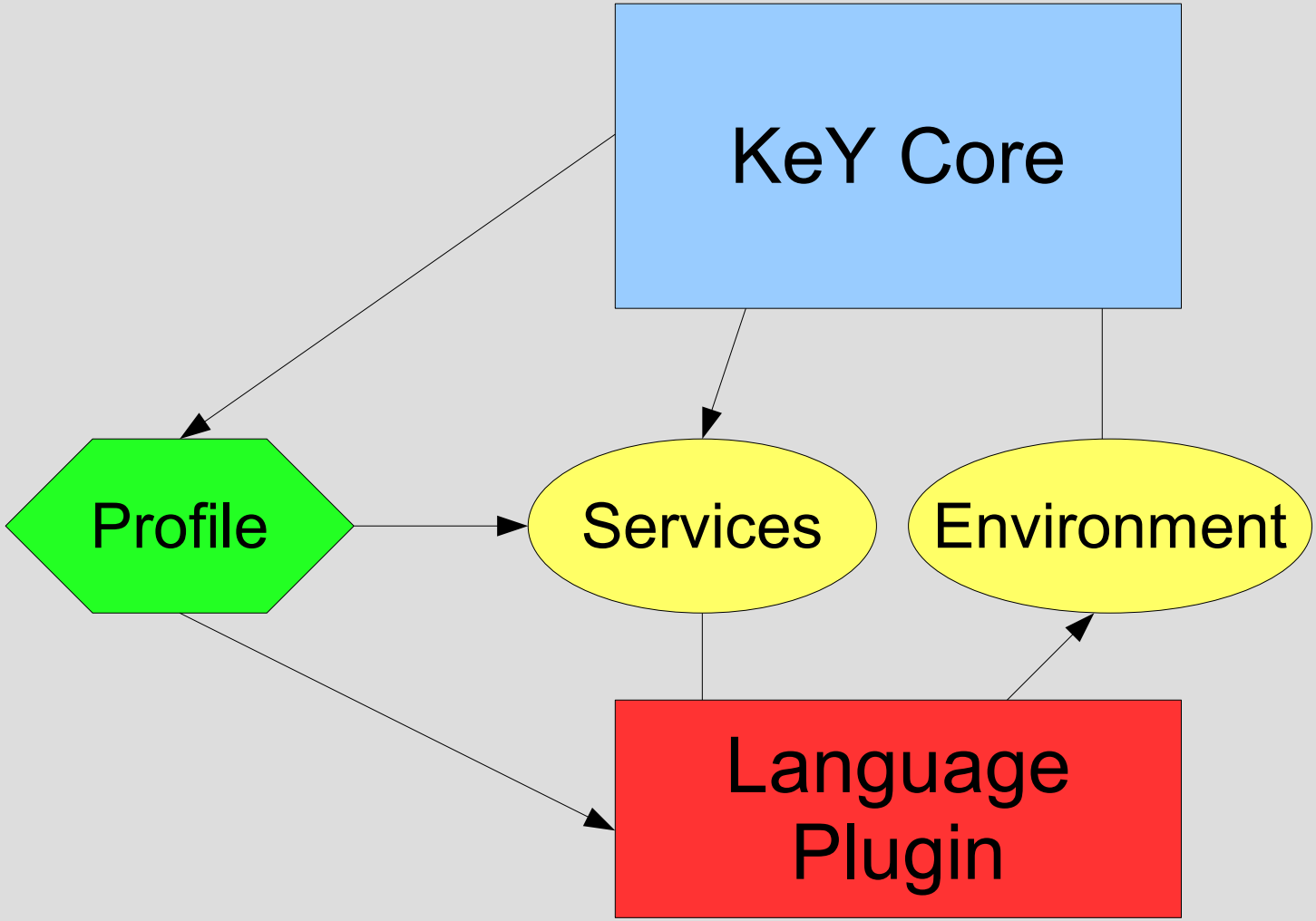


```
arr = (int*)calloc(count, sizeof(int));
if (arr != (int*)0) {
      struct Node* nptr = ifirst;
      int index = 0;
      while (index < count) {
            arr[index] = nptr->elem;
            struct Node* oldPtr = nptr;
            nptr = nptr->next;
            free(oldPtr);
            index++;
      }
}
```

And Now...

KeY-C Demo

KeY Language Variability



KeY Language Variability

- Logic
  - Type System
  - Signature
- Program
  - Type System
  - AST, Parser, Printer
  - Visitors-- (Walkers)
  - Matching
  - Instantiation
- Calculus
  - Rules, Strategy

Further Work

- Language features
  - Function calls, pointers
  - Translation units
  - Jump statements
- Heap Models
  - Unions
  - Deep Copy
  - Untyped Memory Access
- Static Analysis of Aliasing

Sources of Information

- KeY Project: http://www.key-project.org
- Oleg Mürk. *Deductive Verification of C Programs in KeY*. MSc thesis.
- Oleg Mürk, Daniel Larsson, Reiner Hähnle. *A Dynamic Logic for Deductive Verification of C Programs with KeY-C*. C/C++ Verification Workshop at IFM 2007.
- Oleg Mürk, Daniel Larsson, Reiner Hähnle. *KeY-C: A Tool for Verification of C Programs*. Conference on Automated Deduction (CADE-21).
- My home page: http://oleg.myrk.name

Thank You!