Orc and ACL2

Nathan Wetzler
nwetzler@cs.utexas.edu

University of Texas, Austin

May 4, 2008
Outline

Orc
- Problem
- Overview of Orc
- Sites
- Combinators
  - Parallel Combinator
  - Sequential Combinator
  - Pruning Combinator
- Fundamental Sites
- Functions

ACL2
- Motivation
- ACL2 Definition
- Example - Definition
- Example - Theorems
- Industrial-strength Utility
ORC
Parallel/Distributed programming is hard

- Manage threads
- Handle external resources (delays, unreliability, etc)
Problem

Parallel/Distributed programming is hard

- Manage threads
- Handle external resources (delays, unreliability, etc)

Design Pattern

1. Acquire data from an remote service
2. Perform some calculation
3. Invoke other remote services with the results
Overview of Orc

High-level Syntax

- Sites are basic services or components
- Three concurrency combinators integrate sites
- New concepts are implemented with sites and combinators

Structure of an Orc Program

An Orc program has:

- a goal expression
- a set of definitions

The goal expression:

- calls sites
- publishes values
Sites

- Nearly everything is a site in Orc
- Sites look and feel like functions BUT they are not functions
  - Sites can be local or remote
  - Remote sites can be unreliable, i.e. they may not respond to your call
  - Sites return AT MOST one value
- Basic infix operators like +, -, % are actually sites in Orc
- The value returned from a site is “published”
Sites

Sites

- Nearly everything is a site in Orc
- Sites look and feel like functions BUT they are not functions
  - Sites can be local or remote
  - Remote sites can be unreliable, i.e. they may not respond to your call
  - Sites return AT MOST one value
- Basic infix operators like +, -, % are actually sites in Orc
- The value returned from a site is “published”

Examples

Prompt("What is your name?")
Sites

Sites

- Nearly everything is a site in Orc
- Sites look and feel like functions BUT they are not functions
  - Sites can be local or remote
  - Remote sites can be unreliable, i.e. they may not respond to your call
  - Sites return AT MOST one value
- Basic infix operators like +, -, % are actually sites in Orc
- The value returned from a site is “published”

Examples

```
Prompt("What is your name?")
```

```
5 + 3
```
Parallel (or Symmetric) Combinator

- Written as: $f \mid g$
- Executes $f$ and $g$ in parallel
- No direct communication between $f$ and $g$
- Can publish 0, 1, or 2 values
Combinators

Parallel (or Symmetric) Combinator

- Written as: \( f \mid g \)
- Executes \( f \) and \( g \) in parallel
- No direct communication between \( f \) and \( g \)
- Can publish 0, 1, or 2 values

Example

```
Prompt("Name?") \mid Prompt ("Age?")
```
Sequential Combinator

- Written as: \( f >x> g \)
- Executes \( f \) passing any results as \( x \) to \( g \)
- Calls \( g \) as results become available
- Only publishes the values of \( g \)
Combinators

Sequential Combinator

- Written as: \( f >\times> g \)
- Executes \( f \) passing any results as \( x \) to \( g \)
- Calls \( g \) as results become available
- Only publishes the values of \( g \)

Examples

```
Prompt("Name?") >\times> "Hello, " + x
```
Combinators

Sequential Combinator

- Written as: \( f >x> g \)
- Executes \( f \) passing any results as \( x \) to \( g \)
- Calls \( g \) as results become available
- Only publishes the values of \( g \)

Examples

\[
\text{Prompt("Name?") } >x> \text{ "Hello, " + x}
\]

\[
(\text{Prompt("Name?") } | \text{ Prompt("Alias?")}) >x> \text{ "Hello, " + x}
\]
Combinators

Pruning Combinator

- Used for selection, blocking, or termination
- Written as: \( f \ <x< g \)
- Executes \( f \) and \( g \) in parallel
- Any site in \( f \) that needs \( x \) is suspended
- The first result from \( g \) is passed as \( x \) to \( f \). Then \( g \) is terminated
Combinators

Pruning Combinator

- Used for selection, blocking, or termination
- Written as: f <x< g
- Executes f and g in parallel
- Any site in f that needs x is suspended
- The first result from g is passed as x to f. Then g is terminated

Example

"Hello, " + x

<x<

Prompt("Name?") | Prompt("Alias?")
Fundamental Sites

- if
- Rtimer
- stop
Fundamental Sites

- if
- Rtimer
- stop

Example

if(3 == 4) >> "this will never print"
Fundamental Sites

- if
- Rtimer
- stop

Example

```plaintext
if(3 == 4) >> "this will never print"

"immediately"
| Rtimer(3000) >> "...three seconds later..."
| Rtimer(5000) >> "...five seconds later..."
```
Fundamental Sites

- if
- Rtimer
- stop

Example

```plaintext
if(3 == 4) >> "this will never print"

"immediately"
| Rtimer(3000) >> "...three seconds later..."
| Rtimer(5000) >> "...five seconds later..."

Prompt("I’m not listening...") >> stop
```
Functions

Functions will begin evaluation immediately
Can publish as many values as it wants
Both functions and sites can be first-class objects
Functions

- Functions will begin evaluation immediately
- Can publish as many values as it wants
- Both functions and sites can be first-class objects

Example

```python
def nfib(n) = if (n <= 1)
    then 1
    else nfib(n-2) + n1 <n1< nfib(n-1)
```
Professors at UT
Dr. William Cook
Dr. Jayadev Misra

URL
http://orc.csres.utexas.edu/
ACL2
Motivation

Verification

▶ How can you verify that an implementation does what you wanted to?
▶ Specifically, how do you know if your implementation has a bug?
▶ You’d like a “proof of correctness” over all possible input
What is ACL2?

- A Computational Logic for Applicative Common LISP
- Functional subset of LISP
- First-order propositional logic
- Semi-automated
Example

Definition of app:

(defun app (x y)
  (if (consp x)
      (cons (car x) (app (cdr x) y))
      y))
Example

Theorem:

( defthm app-nil-l
  (equal (app nil x) x) )
Example

Theorem:

```
(defthm app-associative
  (equal (app (app a b) c)
         (app a (app b c))))
```
Example

Theorem

(defthm app-nil-r
  (equal (app x nil) x))
Example

Theorem???

(defthm app-nil-r
  (implies (consp x)
    (equal (app x nil) x)))
Example

Theorem:

\(\text{(defthm app-nil-r} \)
\(\text{  (implies (true-listp x)} \)
\(\text{  (equal (app x nil) x)))} \)
Uses of ACL2

- Highly used in hardware design
- I believe that Intel, AMD, Centaur all use ACL2 at some level of their verification process
- Won the 2005 ACM Software System Award (other winners are make, Java, Apache, Tcl/Tk, WWW)
Professors at UT
Dr. Warren Hunt
Dr. J Moore

URL
http://www.cs.utexas.edu/~moore/acl2/