Introduction to Rippling

John Erickson
Dept. of Computer Science
University of Texas at Austin

October 10, 2007
Outline

- What is Rippling?
- Rippling-Out
- Rippling-In
- Rippling-Sideways
- Rippling-Across
- Conclusions
What is Rippling?

- A way to control rewriting during induction
- Based on “rippling-out” the differences between the IH and IC
- Extended to achieve other rewriting goals
- Uses annotated rewrite rules to guide rewriting and ensure termination
Rippling-Out

- IH $x + (y + z) = (x + y) + z$
- IC $s(x) + (y + z) = (s(x) + y) + z$
- $s(...)$ is a wave-front
- $x$ is a wave-hole
- $x + (y + z) = (x + y) + z$ is the skeleton
Wave Rules

- General Form: \( \eta(\xi(\mu)) \rightarrow \zeta(\eta(\mu)) \)
- \( s(U) + V \rightarrow s(U + V) \)
- \( s(U) \times V \rightarrow U \times V + V \)
- \( \text{even}(s(s(U))) \rightarrow \text{even}(U) \)
- \( U + (V + W) \rightarrow (U + V) + W \)
- \( (U + V) + W \rightarrow U + (V + W) \)
Rippling-Out Example

- IH \( x + (y + z) = (x + y) + z \)
- IC \( s(x) + (y + z) = (s(x) + y) + z \)
- \( s(x + (y + z)) = s(x + y) + z \)
- \( s(x + (y + z)) = s((x + y) + z) \)
Rippling-In

► Useful for when one side of an equality is missing a wave rule

► IH $\text{half}(x + x) = x$

► IC $\text{half}(s(x) + s(x)) = s(x)$

► $\text{half}(s(x + s(x))) = s(x)$

► Missing: $U + s(V) \to s(U + V)$

► XF $\text{half}(s(x + s(x))) = s(\text{half}(x + x))$

► $\text{half}(s(x + s(x))) = \text{half}(s(s(x + x)))$

► $x + s(x) = s(x + x)$
Rippling-Sideways

- Unmeasured (free) induction variables can be used as “sinks”
- Rippling-sideways Attempts to ripple wave-front into a sink
- General form: \( \eta(\xi(\mu)^\uparrow, \nu) \rightarrow \eta(\mu, \zeta(nu)^\downarrow) \)
- IH \( \text{rev}(l) <> M = q\text{rev}(l, M) \)
- IC \( \text{rev}(h :: l)^\uparrow <> [m] = q\text{rev}(h :: l)^\uparrow, [m]) \)
- \( \text{rev}(h :: l)^\uparrow <> [m] = q\text{rev}(l, h :: m)^\downarrow ) \)
- \( \text{rev}(l) <> (h :: \text{nil})^\uparrow <> [m] = q\text{rev}(l, h :: m)^\downarrow ) \)
- \( \text{rev}(l) <> [(h :: \text{nil}) <> m]^\downarrow ) = q\text{rev}(l, h :: m)^\downarrow ) \)
Rippling-Across

- Adapts rippling to destructor inductions
- \( U + V = \begin{cases} V & \text{if } U = 0 \\ s(p(U) + V) & \text{else} \end{cases} \)
- Creational Rule: \( U \neq 0 \iff U + V \to s\left( p(U) + V \right) \)

\[
\begin{align*}
p(x) + (y + z) &= (p(x) + y) + z \\ s(p(x) + (y + z)) &= s(p(x) + y) + z
\end{align*}
\]

\[p(x) + (y + z) = (p(x) + y) + z \implies s(p(x) + (y + z)) = s(p(x) + y) + z\]
Conclusions

- **Pros:** Terminating rewrites, ability to use rules right to left, goal directed
- **Cons:** A little bit complicated
- Doesn’t address generalization, lemma generation, or choosing induction scheme
- Some techniques address these problems by asking “how can I make this choice so that rippling will be facilitated?”
- Formalizes informal strategies for rewriting