Forward

Slurp

(foo bar |) (baz) —C-)→ (foo bar |(baz))

Barf

(foo bar |(baz)) —C-}→ (foo bar |)(baz)

Backward

Slurp

(foo) (bar |baz) —C-(→ ((foo) bar |baz)

Barf

((foo) bar |baz) —C-{→ (foo) (bar |baz)

split

(hello | world) —M-S→ (hello) (world)

join

(hello) |(world) —M-J→ (hello | world)

wrap-around

(foo | bar baz) —M-(→ (foo (| bar) baz)

splice

(foo (bar | baz) quux) —M-s→ (foo bar | baz quux)

(a b | c d) →((a b (|) c d)

(a b | c d) →) (a b c d) |

(a b c d) | →<backspace> (a b c |)

| (a b c d) →<delete> (| b c d)

()| →<backspace> |

|() →<delete> (|) →<delete> |
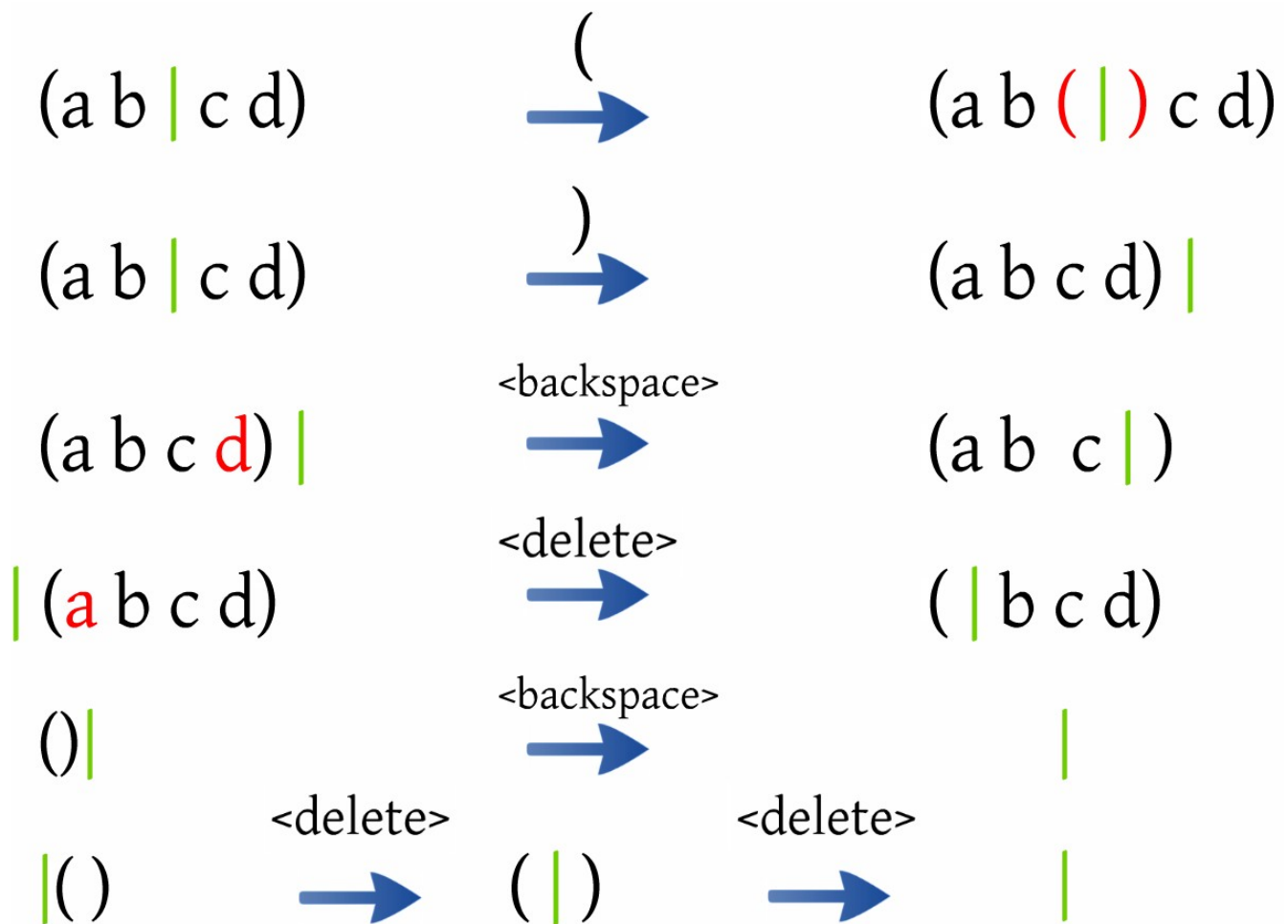
splice killing backward M-<up>

(foo (let [x 5] |(sqrt n)) bar) ⟶ (foo |(sqrt n) bar)

splice killing forward M-<down>

(a (b c |d e) f) ⟶ (a b c| f)