Errata for the first print edition of

Fundamental Proof Methods in Computer Science

MIT Press, 2017, First Edition

Konstantine Arkoudas David Musser

p. 32 *see also Section 2.16* ⟶ *see also Section 2.12*

p. 108 (`map`) *and produces the list* `[(`f V_1`)` $\cdots$ `(`f V_1`)]` ⟶ *and produces the list* `[(`f V_1`)` $\cdots$ `(`f V_n`)]`

p. 108 (`drop`) *greater than the length of* L ⟶ *greater than or equal to the length of* L

p. 123 *as well as the Substitution Axiom* ⟶ *as well as the Functional Substitution axiom*

p. 147 *one of the proofs on page 142*: ⟶ *one of the proofs on page 143*

p. 147 *or the proof on page 145*: ⟶ *or the proof on page 146*

p. 188 (Exercise 3.30) *as follows and prove it:* ⟶ *as follows:*

p. 205 *If the conditional* `(`$\overline{p}$ `==> false)` *is in the assumption base, then*

`(!by-contradiction` p `(`$\overline{p}$ `==> false))`

will produce the conclusion p ⟶
If p *and* q *are complements and the conditional* `(`q `==> false)` *is in the assumption base, then* `(!by-contradiction` p `(`q `==> false))` *will produce the conclusion* p

p. 215 (Disjunctive syllogism) *of the form* `(`p `|` q`)` *and* $\overline{p}$ *(recall that* $\overline{p}$ *is the complement of* p*),* ⟶ *of the form* `(`p `|` q`)` *and* r*, where* p *and* r *are complementary,*

p. 243 *which is to say that every interpretation falsifies* p*. Hence* p *is unsatisfiable.* ⟶ *which is to say that every interpretation falsifies* `(~` p`)`*. Hence* `(~` p`)` *is unsatisfiable.*

p. 270 (footnote 28) *see Section 2.6* ⟶ *see Exercise 2.6*

p. 351 (Quantifier distribution) *and finally, from a premise of the form* $(\forall\, x\ .\ p_1) \vee (\forall\, x\ .\ p_1)$ ⟶ *and finally, from a premise of the form* $(\forall\, x\ .\ p_1) \vee (\forall\, x\ .\ p_2)$

p. 356 (Footnote 15) *role inside* D_2 ⟶ *role in the derivation of the second subgoal*

p. 357 (Second paragraph) *(recall that we write* $\overline{q}$ *for the complement of* q*):* ⟶ *:*

p. 366 (first code listing) `uspec` ⟶ `uspec*`

p. 369 (last code listing) `T = [a b]` ⟶ `T = {a, b}`

p. 414 (last paragraph) *The second and third steps of this example* ⟶ *The steps*

p. 415 (first paragraph) *the first element* ⟶ *the starting element*

p. 487 *develop the integers as a datatype* ⟶ *develop the integers as an inductive structure*

p. 504 *the inference on line 14* ⟶ *the inference on line 13*

p. 577 *by the meta-identifiers* `'when` ⟶ *by the meta-identifier* `'when`

p. 768 *the expression constructorsaccept* ⟶ *the expression constructors accept*

p. 782 *as described in footnote 21* ⟶ *as described in footnote 21 of Chapter 3*

p. 785 *keep in mind that* `I` *returns* ⟶ *keep in mind that* `I'` *returns*

p. 785 *the recursive invocation of* `I` *on either* ⟶ *the recursive invocation of* `I'` *on either*

p. 813 *two arbitrary states* s_1 *and* s_2*, on line 2* ⟶ *two arbitrary states* s_1 *and* s_2*, on line 3*

p. 814 *in which case the call on line 12* ⟶ *in which case the call on line 15*

p. 869 *A deduction* **`pick-witness`** I **`for`** F I_2 D ⟶ *A deduction* **`pick-witness`** I_1 **`for`** F I_2 D

p. 872 *whose sort is an instance of the datatype sort* S_D ⟶ *whose sort is an instance of the inductive sort* S_D

p. 873 *the reflexive constructor* `App` ⟶ *the reflexive constructor* `app`

p. 877 (Case 2) *if* V *is a term variable of sort* T ⟶ *if* V *is a term variable of the same name and of sort* T

p. 877 (At the very end of footnote 18) ⟶ *This was written as* $\widehat{\tau}(S)$ *in Section 2.8, but we use the simpler notation here.*

p. 903 *produced the previous steps* ⟶ *produced by the previous steps*

p. 912 *The syntax of deductions is specified in Appendix A.2* ⟶ *The syntax of deductions is specified in Figure A.2*