Appendix A

Isabelle/Isar quick reference

A.1 Proof commands

A.1.1 Main grammar

main = **notepad begin** *statement** **end**
| **theorem** *name*: *props* **if** *name*: *props* **for** *vars proof*
| **theorem** *name*:
fixes *vars*
assumes *name*: *props*
shows *name*: *props proof*
| **theorem** *name*:
fixes *vars*
assumes *name*: *props*
obtains (*name*) *vars* **where** *props* | ... *proof*

proof = *refinement** *proper_proof*

refinement = **apply** *method*
| **supply** *name* = *thms*
| **subgoal premises** *name* **for** *vars proof*
| **using** *thms*
| **unfolding** *thms*

proper_proof = **proof** *method*$^?$ *statement** **qed** *method*$^?$
| **done**

statement = **{** *statement** **}**
| **next**
| **note** *name* = *thms*
| **let** *term* = *term*
| **write** *name* (*mixfix*)
| **fix** *vars*
| **assume** *name*: *props* **if** *props* **for** *vars*
| **then**$^?$ *goal*

goal = **have** *name*: *props* **if** *name*: *props* **for** *vars proof*
| **show** *name*: *props* **if** *name*: *props* **for** *vars proof*

A.1.2 Primitives

fix x	augment context by $\bigwedge x.\ \square$
assume a: A	augment context by $A \Longrightarrow \square$
then	indicate forward chaining of facts
have a: A	prove local result
show a: A	prove local result, refining some goal
using a	indicate use of additional facts
unfolding a	unfold definitional equations
proof m_1 ... **qed** m_2	indicate proof structure and refinements
{ ... **}**	indicate explicit blocks
next	switch proof blocks
note $a = b$	reconsider and declare facts
let $p = t$	abbreviate terms by higher-order matching
write c (mx)	declare local mixfix syntax

A.1.3 Abbreviations and synonyms

by m_1 m_2	$\equiv$	**proof** m_1 **qed** m_2
..	$\equiv$	**by** *standard*
.	$\equiv$	**by** *this*
from a	$\equiv$	**note** a **then**
with a	$\equiv$	**from** a **and** *this*
from *this*	$\equiv$	**then**

A.1.4 Derived elements

also$_0$	$\approx$	**note** *calculation* = *this*
also$_{n+1}$	$\approx$	**note** *calculation* = *trans* [*OF calculation this*]
finally	$\approx$	**also from** *calculation*
moreover	$\approx$	**note** *calculation* = *calculation this*
ultimately	$\approx$	**moreover from** *calculation*
presume a: A	$\approx$	**assume** a: A
define x **where** $x = t$	$\approx$	**fix** x **assume** x_def: $x = t$
consider x **where** $A \mid \ldots$	$\approx$	**have** *thesis* **if** $\bigwedge x.\ A \Longrightarrow$ *thesis* **and** ... **for** *thesis*
obtain x **where** a: A $\langle proof \rangle$	$\approx$	**consider** x **where** A $\langle proof \rangle$ **fix** x **assume** a: A
case c	$\approx$	**fix** x **assume** c: A
sorry	$\approx$	**by** *cheating*

A.1.5 Diagnostic commands

typ τ	print type
term t	print term
prop φ	print proposition
thm a	print fact
print_statement a	print fact in long statement form

A.2 Proof methods

Single steps (forward-chaining facts)

assumption	apply some goal assumption
this	apply current facts
rule a	apply some rule
standard	apply standard rule (default for **proof**)
contradiction	apply ¬ elimination rule (any order)
cases t	case analysis (provides cases)
induct x	proof by induction (provides cases)

Repeated steps (inserting facts)

–	no rules
intro a	introduction rules
intro_classes	class introduction rules
intro_locales	locale introduction rules (without body)
unfold_locales	locale introduction rules (with body)
elim a	elimination rules
unfold a	definitional rewrite rules

Automated proof tools (inserting facts)

iprover	intuitionistic proof search
blast, *fast*	Classical Reasoner
simp, *simp_all*	Simplifier (+ Splitter)
auto, *force*	Simplifier + Classical Reasoner
arith	Arithmetic procedures

A.3 Attributes

Rules

OF a	rule resolved with facts (skipping "_")
of t	rule instantiated with terms (skipping "_")
where x = t	rule instantiated with terms, by variable name
symmetric	resolution with symmetry rule
THEN b	resolution with another rule
rule_format	result put into standard rule format
elim_format	destruct rule turned into elimination rule format

Declarations

simp	Simplifier rule
intro, *elim*, *dest*	Pure or Classical Reasoner rule
iff	Simplifier + Classical Reasoner rule
split	case split rule
trans	transitivity rule
sym	symmetry rule

A.4 Rule declarations and methods

	rule	*iprover*	*blast* *fast*	*simp* *simp_all*	*auto* *force*
Pure.elim! *Pure.intro*!	×	×			
Pure.elim Pure.intro	×	×			
elim! *intro*!	×		×		×
elim intro	×		×		×
iff	×		×	×	×
iff?	×				
elim? intro?	×				
simp				×	×
cong				×	×
split				×	×

A.5 Proof scripts

A.5.1 Commands

apply *m*	apply proof method during backwards refinement
apply_end *m*	apply proof method (as if in terminal position)
supply *a*	supply facts during backwards refinement
subgoal	nested proof during backwards refinement
defer *n*	move subgoal to end
prefer *n*	move subgoal to start
back	backtrack last command
done	complete proof

A.5.2 Methods

rule_tac insts	resolution (with instantiation)
erule_tac insts	elim-resolution (with instantiation)
drule_tac insts	destruct-resolution (with instantiation)
frule_tac insts	forward-resolution (with instantiation)
cut_tac insts	insert facts (with instantiation)
thin_tac φ	delete assumptions
subgoal_tac φ	new claims
rename_tac x	rename innermost goal parameters
rotate_tac n	rotate assumptions of goal
tactic text	arbitrary ML tactic
case_tac t	exhaustion (datatypes)
induct_tac x	induction (datatypes)
ind_cases t	exhaustion + simplification (inductive predicates)