Appendix A

# Isabelle/Isar quick reference

## A.1 Proof commands

### A.1.1 Primitives and basic syntax

| | |
|---|---|
| **fix** $x$ | augment context by $\bigwedge x.\ \square$ |
| **assume** $a$: $\varphi$ | augment context by $\varphi \Longrightarrow \square$ |
| **then** | indicate forward chaining of facts |
| **have** $a$: $\varphi$ | prove local result |
| **show** $a$: $\varphi$ | 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 blocks |
| **note** $a = b$ | reconsider facts |
| **let** $p = t$ | abbreviate terms by higher-order matching |
| **write** $c$ $(mx)$ | declare local mixfix syntax |

*proof* = *prfx*$^*$ **proof** *method*$^?$ *stmt*$^*$ **qed** *method*$^?$
| *prfx*$^*$ **done**

*prfx* = **apply** *method*
| **using** *facts*
| **unfolding** *facts*

*stmt* = **{** *stmt*$^*$ **}**
| **next**
| **note** *name* = *facts*
| **let** *term* = *term*
| **write** *name* (*mixfix*)
| **fix** *var*$^+$
| **assume** *name*: *props*
| **then**$^?$ *goal*

*goal* = **have** *name*: *props proof*
| **show** *name*: *props proof*

### A.1.2 Abbreviations and synonyms

| | | |
|---|---|---|
| **by** $m_1$ $m_2$ | $\equiv$ | **proof** $m_1$ **qed** $m_2$ |
| **..** | $\equiv$ | **by** *rule* |
| **.** | $\equiv$ | **by** *this* |
| **hence** | $\equiv$ | **then have** |
| **thus** | $\equiv$ | **then show** |
| **from** $a$ | $\equiv$ | **note** $a$ **then** |
| **with** $a$ | $\equiv$ | **from** $a$ **and** *this* |
| **from** *this* | $\equiv$ | **then** |
| **from** *this* **have** | $\equiv$ | **hence** |
| **from** *this* **show** | $\equiv$ | **thus** |

### A.1.3 Derived elements

| | | |
|---|---|---|
| $\textbf{also}_0$ | $\approx$ | **note** *calculation* = *this* |
| $\textbf{also}_{n+1}$ | $\approx$ | **note** *calculation* = *trans* [*OF calculation this*] |
| **finally** | $\approx$ | **also from** *calculation* |
| **moreover** | $\approx$ | **note** *calculation* = *calculation this* |
| **ultimately** | $\approx$ | **moreover from** *calculation* |
| **presume** $a$: $\varphi$ | $\approx$ | **assume** $a$: $\varphi$ |
| **def** $a$: $x \equiv t$ | $\approx$ | **fix** $x$ **assume** $a$: $x \equiv t$ |
| **obtain** $x$ **where** $a$: $\varphi$ | $\approx$ | ... **fix** $x$ **assume** $a$: $\varphi$ |
| **case** $c$ | $\approx$ | **fix** $x$ **assume** $c$: $\varphi$ |
| **sorry** | $\approx$ | **by** *cheating* |

### A.1.4 Diagnostic commands

| | |
|---|---|
| **print_state** | print proof state |
| **print_statement** | print fact in long statement form |
| **thm** $a$ | print fact |
| **prop** $\varphi$ | print proposition |
| **term** $t$ | print term |
| **typ** $\tau$ | print type |

## A.2 Proof methods

**Single steps (forward-chaining facts)**

| | |
|---|---|
| *assumption* | apply some assumption |
| *this* | apply current facts |
| *rule a* | apply some rule |
| *rule* | 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 |
| *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* | | | | × | × |