DRAFT: Transnumber Derivations and Model

Norbert Völker, University of Essex

October 27, 2006

Contents

1 Transarithmetic: Axiomatic Classes

theory *TransNumberAxclass*

imports *Main Real*

begin

1.1 New constants: xor, infinity, sgn

Note: This "list" xor is different from binary xor when there are three or more arguments.

consts
  xor :: *bool list* ⇒ *bool*
primrec
  xor [] = *False*
  xor (*a* # *x*) = ((*xor x* ∧ ¬ *a*) | (¬ (*list-ex id x*) ∧ *a*))

lemma *xor-singleton*: *xor* [*b*] = *b*
  by *simp*

lemma *xor-pair*: *xor* [*a*,*b*] = (*a* ≠ *b*)
  by *auto*

lemma *xor-triple*:
  xor [*a*,*b*,*c*] = ((*a* ∧ ¬ *b* ∧ ¬*c*) ∨ (¬ *a* ∧ *b* ∧ ¬*c*) ∨ (¬ *a* ∧ ¬ *b* ∧ *c*))
  by *auto*

axclass *infinity* < *type*

Note: there is no separate constant minus-infinity

consts
  infinity :: 'a::*infinity* (∞ *100*)

axclass *nullity* < *type*
  consts
    nullity:: 'a ::*nullity* (Φ)

axclass *sgn* < *zero*,*one*,*minus*, *ord*

consts
  sgn :: 'a :: *sgn* ⇒ 'a

axclass *trans-sgn* < *sgn*, *nullity*
  trans-sgn:
 sgn a = (*if 0* < *a then 1 else*
                if 0 = *a then 0 else*
                if a < *0 then* − *1 else*
                (∗ *a* = Φ ∗) Φ)

1.2 Axiomatic class trans_add

axclass

trans-add < *zero*, *infinity*, *nullity*, *plus*, *minus*
A1: $a + (b + c) = (a + b) + c$
A2: $a + b = b + a$
A3: $0 + a = a$
A4: $\Phi + a = \Phi$
A5: $[\![\ a \neq -\infty;\ a \neq \Phi\ ]\!] \Longrightarrow \infty + a = \infty$

A6: $a - b = a + (-b)$
A7: $-(-a) = a$
A8: $[\![\ a \neq \infty;\ a \neq -\infty;\ a \neq \Phi\ ]\!] \Longrightarrow a - a = 0$
A9: $-\Phi = \Phi$
A10: $[\![\ a \neq \infty;\ a \neq \Phi\ ]\!] \Longrightarrow a - \infty = -\infty$
A11: $\infty - \infty = \Phi$

instance *trans-add* $\subseteq$ *comm-monoid-add*
apply (*intro-classes*)
apply (*rule A1* [*THEN sym*])
by (*rule A2*, *rule A3*)

1.3 Axiomatic class trans_mult (Axioms A12-A28)

axclass
trans-mult < *trans-add*, *trans-sgn*, *one*, *times*, *inverse*, *ord*

A12: $a * (b * c) = (a * b) * c$
A13: $a * b = b * a$
A14: $1 * a = a$

A15: $\Phi * a = \Phi$
A16: $\infty * 0 = \Phi$
A17: $a\ /\ b = a * inverse\ b$
A18: $[\![\ a \neq 0;\ a \neq \infty;\ a \neq -\infty;\ a \neq \Phi\ ]\!] \Longrightarrow a\ /\ a = 1$
A19: $a \neq -\infty \Longrightarrow inverse\ (inverse\ a) = a$
A20: $inverse\ 0 = \infty$
A21: $inverse\ (-\infty) = 0$
A22: $inverse\ \Phi = \Phi$

A23: $(\infty * a = \infty) = (0 < a\ )$
A24: $(\infty * a = -\infty) = (a < 0)$
A25: $0 < \infty$

A26: $(0 < a - b) = (b < a)$
A27: $(a > b) = (b < a)$
A28: $xor\ [a < 0,\ a = 0,\ 0 < a,\ a = \Phi]$

A29: $\neg\ ((a = \infty \vee a = -\infty) \wedge sgn\ b \neq sgn\ c \wedge (b + c \notin \{0,\Phi\}))$
$\Longrightarrow a * (b+c) = (a * b)+(a * c)$

A30: $a \leq b = (a = b \vee a < b)$

instance *trans-mult* $\subseteq$ *comm-monoid-mult*
by (*intro-classes*, *rule A12* [*THEN sym*], *rule A13*, *rule A14*)

1.4 Axiomatic classes trans_complete and trans_reals

constdefs
lattice-complete :: ('*a*::*ord*) *set* $\Rightarrow$ *bool*
lattice-complete xs ==
$\forall$ *ys*. *ys* $\subseteq$ *xs* $\longrightarrow$ ($\exists$ *u* $\in$ *xs*. ($\forall$ *y* $\in$ *ys*. *y* $\leq$ *u*)
$\wedge$ ($\forall$ *v* $\in$ *xs*. ($\forall$ *y* $\in$ *ys*. *y* $\leq$ *v*) $\longrightarrow$ *u* $\leq$ *v*))

axclass *trans-complete* < *trans-add*, *one*, *times*, *inverse*, *ord*
A31: *lattice-complete* {*x*. *x* $\neq$ Φ}

axclass *trans-reals* < *trans-mult*, *trans-complete*

TODO: validate definition by proving lattice-completeness of [0..1]

end

2 Transarithmetic Theorems

theory *TransNumberDerivations*

imports *TransNumberAxclass*

begin

declare *A3*[*simp*] *A4*[*simp*] *A7*[*simp*] *A9*[*simp*] *A11*[*simp*]
A14[*simp*] *A15*[*simp*] *A16*[*simp*]
A20[*simp*] *A21*[*simp*] *A22*[*simp*] *A25*[*simp*]

constdefs
reals :: ('*a*::*trans-reals*) *set*

reals == $\{x.\ x \neq \Phi \wedge x \neq \infty \wedge x \neq -\infty\}$

Note : following subclassing allows reuse of standard add_ac, etc.

instance *trans-reals* $\subseteq$ *comm-monoid-add* **..**

2.1 Very elementary equations

lemma *additive-identity-right*[*simp*]: $(x :: 'a :: trans\text{-}reals) + 0 = x$
by (*subst A2*, *simp*)

lemma *multiplicative-identity-right*[*simp*]: $(x :: 'a :: trans\text{-}reals) * 1 = x$
by (*subst A13*, *simp*)

lemma *additive-nullity-right*[*simp*]: $(x :: 'a :: trans\text{-}reals) + \Phi = \Phi$
by (*subst A2*, *simp*)

lemma *additive-infinity-right*:$[\![x \neq -\infty; x \neq \Phi]\!] \Longrightarrow (x :: 'a :: trans\text{-}reals) + \infty = \infty$
by (*subst A2*, *subst A5*, *auto*)

lemma *minus-minus*[*simp*]: $(x :: 'a :: trans\text{-}reals) - - y = x + y$
by (*simp add*: *A6*)

lemma *zero-mult-infinity*[*simp*]:
$0 * (\infty :: 'a :: trans\text{-}reals) = \Phi$
by (*subst A13*, *rule A16*)

lemma *nullity-minus-left*[*simp*]: $\Phi - (x :: 'a :: trans\text{-}reals) = \Phi$
by (*simp add*: *A6*)

lemma *nullity-minus-right*[*simp*]: $(x :: 'a :: trans\text{-}reals) - \Phi = \Phi$
by (*simp add*: *A6*)

lemma *zero-minus-eq-uminus*[*simp*]: $(0 :: 'a :: trans\text{-}reals) - x = -x$
by (*simp add*: *A6*)

lemma *uminus-eq-uminus*[*simp*]: $(-(x :: 'a :: trans\text{-}reals) = -y) = (x = y)$
by (*safe*, *drule-tac* $f{=}\lambda\ x.\ -\ x$ **in** *arg-cong*, *simp*)

lemma *uminus-add-eq-minus*: $-x + y = y - (x :: 'a :: trans\text{-}reals)$
by (*subst A2*, *simp add*: *A6*)

lemma *uminus-minus-commute*: $-x - y = - y - (x :: 'a :: trans\text{-}reals)$
by (*simp add*: *A6 A2*)

lemma *x-add-y-minus-y*:
$[\![y \neq \Phi; y \neq \infty; y \neq -\infty]\!] \Longrightarrow x + y - y = (x :: 'a :: trans\text{-}reals)$
apply (*subst A6*, *subst A1*[*THEN sym*])
by (*simp add*: *A6*[*THEN sym*] *A8*)

lemma *uminus-x-add-x*: $[\![x \neq \infty; x \neq -\infty; x \neq \Phi]\!] \Longrightarrow -x + x = (0::'a::trans\text{-}reals)$

by (*subst A2*, *simp add*: *A6*[*THEN sym*] *A8*)

lemma *x-minus-y-add-y*:
$[\![y \neq \Phi; y \neq \infty; y \neq -\infty]\!] \Longrightarrow x - y + y = (x::'a::trans\text{-}reals)$
apply (*subst A6*, *subst A1*[*THEN sym*])
by (*simp add*: *uminus-x-add-x*)

lemma *uminus-eq-nullity-iff*[*simp*]: !! $x::'a::trans\text{-}reals.\ (-x = \Phi) = (x = \Phi)$
by (*auto*, *drule-tac* $f{=}\lambda\ x.\ -x$ **in** *arg-cong*, *simp*)

lemma *uminus-eq-infinity-iff*[*simp*]: !! $x::'a::trans\text{-}reals.\ (-x = \infty) = (x = -\infty)$
by (*auto*, *drule-tac* $f{=}\lambda\ x.\ -x$ **in** *arg-cong*, *simp*)

lemma *infinity-minus*: $[\![x \neq \Phi; x \neq \infty]\!] \Longrightarrow \infty - x = (\infty::'a::trans\text{-}reals)$
by (*subst A6*, *subst A5*, *simp-all*)

2.2 Distinctness of six basic constants: 0, nullity, +1,-1, plus/minus infinity

lemma *not-zero-less-zero*[*simp*]: $\neg\ 0 < (0::'a::trans\text{-}reals)$
by (*cut-tac* $a{=}0::'a$ **in** *A28*, *auto*)

lemma *zero-noteq-nullity*[*simp*]: $(0::'a::trans\text{-}reals) \neq \Phi$
by (*cut-tac* $a{=}\Phi::'a$ **in** *A28*, *auto*)

lemmas *nullity-noteq-zero*[*simp*] = *zero-noteq-nullity*[*THEN not-sym*]

lemma *zero-noteq-infinity*[*simp*]: $(0::'a::trans\text{-}reals) \neq \infty$
by (*cut-tac* $a{=}(\infty::'a)$ **in** *A28*, *simp*)

lemmas *infinity-noteq-zero*[*simp*] = *zero-noteq-infinity*[*THEN not-sym*]

lemma *nullity-not-less-zero*[*simp*]: $\neg\ \Phi < (0::'a::trans\text{-}reals)$
by (*cut-tac* $a{=}(\Phi::'a)$ **in** *A28*, *simp*)

lemma *zero-not-less-nullity*[*simp*]: $\neg\ 0 < (\Phi::'a::trans\text{-}reals)$
by (*cut-tac* $a{=}(\Phi::'a)$ **in** *A28*, *simp*)

lemma *infinity-noteq-nullity*[*simp*]: $(\infty::'a::trans\text{-}reals) \neq \Phi$
by (*rule notI*, *drule-tac* $f{=}\lambda\ x.\ 0 < x$ **in** *arg-cong*, *simp*)

lemmas *nullity-noteq-infinity*[*simp*] = *infinity-noteq-nullity*[*THEN not-sym*]

lemma *zero-less-one*[*simp*]: $(0::'a::trans\text{-}reals) < 1$
by (*simp add*: *A23*[*THEN sym*])

lemma *not-one-less-zero*[*simp*]: $\neg\ (1 < (0::'a::trans\text{-}reals))$
by (*cut-tac* $a{=}(1::'a)$ **in** *A28*, *auto*)

lemma *zero-noteq-one*[*simp*]: $(0::'a::trans\text{-}reals) \neq 1$
by (*cut-tac* $a{=}(1::'a)$ **in** *A28*, *auto*)

lemmas *one-noteq-zero*[*simp*] = *zero-noteq-one*[*THEN not-sym*]

lemma *one-noteq-infinity*[*simp*]: $(1::'a::trans\text{-}reals) \neq \infty$
apply (*rule classical*, *simp*)
by (*drule-tac* $f{=}\lambda\ y.\ y * 0$ **in** *arg-cong*, *simp*)

lemmas *infinity-noteq-one*[*simp*] = *one-noteq-infinity* [*THEN not-sym*]

lemma *nullity-noteq-one*[*simp*]: $\Phi \neq (1::'a::trans\text{-}reals)$
by (*rule notI*, *drule-tac* $f{=}\lambda\ x.\ 0 < x$ **in** *arg-cong*, *simp*)

lemmas *one-not-nullity* [*simp*] = *nullity-noteq-one* [*THEN not-sym*]

lemma *minus-one-less-zero*[*simp*]: $(-\ 1::'a::trans\text{-}reals) < 0$
by (*subst A26*[*THEN sym*], *simp*)

lemma *minus-one-mult-infinity*[*simp*]: $(-\ 1::'a::trans\text{-}reals) * \infty = -\ \infty$
by (*subst A13*, *rule A24*[*THEN iffD2*], *rule minus-one-less-zero*)

lemma *zero-noteq-minus-one*[*simp*]: $(0::'a::trans\text{-}reals) \neq -\ 1$
by (*cut-tac* $a{=}-\ (1::'a)$ **in** *A28*, *auto*)

lemmas *minus-one-noteq-zero*[*simp*] = *zero-noteq-minus-one* [*THEN not-sym*]

lemma *not-one-less-minus-one*[*simp*]: $\neg\ ((0::'a::trans\text{-}reals) < -\ 1)$
by (*cut-tac* $a{=}-\ (1::'a)$ **in** *A28*, *auto*)

lemma *one-noteq-minus-one*[*simp*]: $(1::'a::trans\text{-}reals) \neq -\ 1$
by (*rule notI*, *drule-tac* $f{=}\lambda\ x.\ x < 0$ **in** *arg-cong*, *simp*)

lemmas *minus-one-noteq-one*[*simp*] = *one-noteq-minus-one* [*THEN not-sym*]

lemma *infinity-noteq-minus-one*[*simp*]: $\infty \neq (-\ 1::'a::trans\text{-}reals)$
by (*rule notI*, *drule-tac* $f{=}\lambda\ x.\ 0 < x$ **in** *arg-cong*, *simp*)

lemmas *minus-one-not-infinity* [*simp*] = *infinity-noteq-minus-one* [*THEN not-sym*]

lemma *nullity-noteq-minus-one*[*simp*]: $\Phi \neq (-\ 1::'a::trans\text{-}reals)$
by (*rule notI*, *drule-tac* $f{=}\lambda\ x.\ x < 0$ **in** *arg-cong*, *simp*)

lemmas *minus-one-not-nullity* [*simp*] = *nullity-noteq-minus-one* [*THEN not-sym*]

lemma *zero-noteq-minus-infinity*[*simp*]: $0 \neq (-\infty ::'a::trans\text{-}reals)$
apply (*cut-tac a*=$-$ $(\infty ::'a)$ **in** *A28*, *simp*,*safe*)
apply (*erule notE*)
apply (*rule A26*[*THEN iffD1*])
by (*subst A6*, *subst A7*, *simp*)

lemmas *minus-infinity-noteq-zero*[*simp*] = *zero-noteq-minus-infinity*[*THEN not-sym*]

lemma *uminus-zero*[*simp*]: $-(0 ::'a::trans\text{-}reals) = 0$
by (*cut-tac a*=*0*::*'a* **in** *A8*, *simp-all add*: *A6*)

lemma *uminus-eq-zero-iff*[*simp*]: !! $x::'a::trans\text{-}reals.\ (-x = 0) = (x = 0)$
by (*auto*, *drule-tac f*=$\lambda\ x.\ -x$ **in** *arg-cong*, *simp*)

lemma *minus-infinity-less-zero*[*simp*]: $(-\infty ::'a::trans\text{-}reals) < 0$
by (*rule A26* [*THEN iffD1*], *simp*)

lemma *minus-infinity-mult-infinity*[*simp*]: $(-\infty ::'a::trans\text{-}reals) * \infty = -\infty$
by (*subst A13*, *rule A24*[*THEN iffD2*], *rule minus-infinity-less-zero*)

lemma *not-zero-less-minus-infinity*[*simp*]: $\neg\ 0 <\ (-\infty ::'a::trans\text{-}reals)$
by (*cut-tac a*=$-$ $(\infty ::'a)$ **in** *A28*, *auto*)

lemma *one-noteq-minus-infinity*[*simp*]: $1 \neq (-\infty ::'a::trans\text{-}reals)$
apply (*rule notI*, *drule-tac f*=$\lambda\ x.\ -\ x$ **in** *arg-cong*)
by (*simp only*: *A7* *minus-one-not-infinity*)

lemmas *minus-infinity-noteq-one*[*simp*] = *one-noteq-minus-infinity* [*THEN not-sym*]

lemma *infinity-noteq-minus-infinity*[*simp*]: $\infty \neq (-\infty ::'a::trans\text{-}reals)$
by (*rule notI*, *drule-tac f*=$\lambda\ x.\ 0 < x$ **in** *arg-cong*, *simp*)

lemmas *minus-infinity-noteq-infinity*[*simp*]
= *infinity-noteq-minus-infinity*[*THEN not-sym*]

lemma *nullity-noteq-minus-infinity*[*simp*]: $\Phi \neq (-\infty ::'a::trans\text{-}reals)$
by (*rule notI*, *drule-tac f*=$\lambda\ x.\ -\ x$ **in** *arg-cong*, *simp*)

lemmas *minus-infinity-noteq-nullity*[*simp*]
= *nullity-noteq-minus-infinity*[*THEN not-sym*]

lemma *one-minus-one-eq-zero*[*simp*]: $(1::'a::trans\text{-}reals) - 1 = 0$
by (*subst A8*, *auto*)

lemma *less-irreflexive*[*simp*]: $\neg\ x < (x::'a::trans\text{-}reals)$
apply (*subst A26*[*THEN sym*])

apply (*case-tac* $x = \infty$, *simp*)
apply (*case-tac* $x = -\infty$, *simp add*: *A2 A6*[*THEN sym*])
apply (*case-tac* $x{=}\Phi$, *simp add*: *A6*)
by (*subst A8*, *auto*)

TODO: less_transitive

lemma *not-less-zero-and-zero-less*:
$[\![x < (0 ::'a::trans\text{-}reals);\ 0 < x]\!] \Longrightarrow P$
by (*cut-tac* $a{=}(x::'a)$ **in** *A28*, *auto*)

lemma *less-zero-imp-not-zero*: $x < (0 ::'a::trans\text{-}reals) \Longrightarrow x \neq 0$
by *auto*

lemma *zero-less-imp-not-zero*: $(0 ::'a::trans\text{-}reals) < x \Longrightarrow x \neq 0$
by *auto*

2.3 sign equivalences

lemma *sgn-negative-iff*:
!! $a ::'a::trans\text{-}reals.\ (sgn\ a = -1) = (a < 0)$
apply (*unfold trans-sgn*, *auto*)
by (*erule not-less-zero-and-zero-less*, *auto*)

lemma *sgn-positive-iff*:
!! $a ::'a::trans\text{-}reals.\ (sgn\ a = 1) = (0 < a)$
by (*unfold trans-sgn*, *auto*)

lemma *sgn-zero*[*simp*]: $sgn\ (0 ::'a::trans\text{-}reals) = 0$
by (*unfold trans-sgn*, *simp*)

lemma *sgn-zero-iff*: $(sgn\ (x ::'a::trans\text{-}reals) = 0) = (x = 0)$
by (*unfold trans-sgn*, *simp*)

lemma *sgn-nullity*[*simp*]: $sgn\ (\Phi ::'a::trans\text{-}reals) = \Phi$
by (*unfold trans-sgn*, *simp*)

lemma *sgn-nullity-iff*: $(sgn\ (x ::'a::trans\text{-}reals) = \Phi) = (x = \Phi)$
apply (*unfold trans-sgn*, *simp*)
by (*cut-tac* $a{=}(x::'a)$ **in** *A28*, *auto*)

2.4 Algebraic stuff and inequalities...

lemma *infinity-add-infinity*[*simp*]: $(\infty ::'a::trans\text{-}reals) + \infty = \infty$
by (*simp add*: *A5*)

lemma *minus-infinity-minus-infinity*[*simp*]: $-(\infty ::'a::trans\text{-}reals) - \infty = -\infty$
by (*simp add*: *A10*)

lemma *minus-infinity-add-infinity*[*simp*]: $-(\infty ::'a::trans\text{-}reals) + \infty = \Phi$

by (*subst A2*, *simp add*: *A6*[*THEN sym*])

lemma *not-nullity-less*[*simp*]: $\neg\ \Phi\ <\ (x::'a::trans\text{-}reals)$
apply (*subst A26*[*THEN sym*])
apply (*simp add*: *A23*[*THEN sym*] *A6*)
by (*subst A13*, *simp*)

lemma *not-less-nullity*[*simp*]: $\neg\ x\ <\ (\Phi::'a::trans\text{-}reals)$
apply (*subst A26*[*THEN sym*])
apply (*simp add*: *A23*[*THEN sym*] *A6*)
by (*subst A13*, *simp*)

lemma *not-nullity-le-infinity*: $x \neq \Phi \Longrightarrow x < \infty \mid x = (\infty::'a::trans\text{-}reals)$
by (*subst A26*[*THEN sym*], *clarsimp simp*: *infinity-minus*)

lemma *less-infinity-iff*: $(x < \infty) = (x \neq (\infty::'a::trans\text{-}reals) \wedge x \neq \Phi)$
by (*auto dest*: *not-nullity-le-infinity*)

lemma *not-nullity-minus-infinity-le*: $x \neq \Phi \Longrightarrow\ -\infty\ < x \mid x = (-\infty::'a::trans\text{-}reals)$
by (*subst A26*[*THEN sym*], *clarsimp simp*: *infinity-minus additive-infinity-right*)

lemma *minus-infinity-less-iff*: $(-\infty < x) = (x \neq (-\infty::'a::trans\text{-}reals) \wedge x \neq \Phi)$
by (*auto dest*: *not-nullity-minus-infinity-le*)

lemma *infinity-add-eq-nullity*:
$(\infty + x = \Phi) = (x = -\ (\infty::'a::trans\text{-}reals) \vee x = \Phi)$
apply (*safe*, *simp-all add*: *A6*[*THEN sym*])
apply (*case-tac* $x{=}\infty$, *simp*)
apply (*drule-tac* $f{=}\lambda\ a.\ a - x$ **in** *arg-cong*)
apply (*erule-tac* $Q{=}?x = ?y$ **in** *contrapos-pp*)
apply (*simp add*: *A1*[*THEN sym*] *A6*)
by(*simp add*: *A6*[*THEN sym*] *A8*)

lemma *infinity-minus-eq-nullity*:
$(x - \infty = \Phi) = (x =\ (\infty::'a::trans\text{-}reals) \vee x = \Phi)$
apply (*safe*, *simp-all add*: *A6*[*THEN sym*])
apply (*case-tac* $x{=}-\infty$, *simp*)
apply (*drule-tac* $f{=}\lambda\ a.\ -x + a$ **in** *arg-cong*)
apply (*erule-tac* $Q{=}?x = ?y$ **in** *contrapos-pp*)
by (*simp add*: *A1 A6 uminus-x-add-x*)

lemma *add-eq-nullity-iff*:
$(x + y = (\Phi::'a::trans\text{-}reals)) =$
$(x = \Phi \vee y = \Phi \vee (x{=}\infty \wedge y = -\infty) \vee (x{=} -\infty \wedge y = \infty))$
apply *auto*
apply (*simp-all add*: *A6*[*THEN sym*])

apply (*drule-tac* $f{=}\lambda\ a.\ -x + a$ **in** *arg-cong*)

apply (*erule-tac* Q=*?x* = *?y* **in** *contrapos-pp*)
apply (*simp add*: *A1 uminus-x-add-x*)

apply (*case-tac* x=∞, *simp add*: *infinity-add-eq-nullity*)
apply (*drule-tac* f=λ *a*. − *x* + *a* **in** *arg-cong*)
apply (*erule-tac* Q=*?x* = *?y* **in** *contrapos-pp*)
apply (*simp add*: *A1 uminus-x-add-x*)

apply (*case-tac* y=$-\infty$, *simp add*: *A6*[*THEN sym*] *infinity-minus-eq-nullity*)
apply (*drule-tac* f=λ *a*. *a* − *y* **in** *arg-cong*)
apply (*erule-tac* Q=*?x* = *?y* **in** *contrapos-pp*)
apply (*simp add*: *x-add-y-minus-y*)

apply (*drule-tac* f=λ *a*. *a* − *y* **in** *arg-cong*)
apply (*erule-tac* Q=*?x* = *?y* **in** *contrapos-pp*)
by (*simp add*: *x-add-y-minus-y*)

lemma *minus-nullity-eq-iff*:
(*x* − *y* = (Φ ::'*a*::*trans-reals*)) =
(*x* = Φ ∨ *y* = Φ ∨ (*x*=∞ ∧ *y* = ∞) ∨ (*x*= − ∞ ∧ *y* = −∞))

by (*simp add*: *A6 add-eq-nullity-iff*)

lemma *add-infinity*:
x + (∞ ::'*a*::*trans-reals*) = (*if x* = Φ | *x* = −∞ *then* Φ *else* ∞)
apply (*case-tac* x=Φ, *simp*)
apply (*case-tac* x=$-\infty$, *subst A2*, *simp add*: *A6*[*THEN sym*])
by (*subst A2*, *simp add*: *A5*)

lemma *add-infinity-not-eq*[*simp*]:
(*x* ::'*a*::*trans-reals*) + ∞ ≠ *0* ∧ (*x* ::'*a*::*trans-reals*) + ∞ ≠ −∞
by (*simp add*: *add-infinity*)

lemma *subtract-infinity*:
x − (∞ ::'*a*::*trans-reals*) = (*if x* = Φ | *x* = ∞ *then* Φ *else* −∞)
apply (*case-tac* x=Φ, *simp*)
apply (*case-tac* x=∞, *simp*)
by (*simp add*: *A10*)

lemma *subtract-infinity-not-eq*[*simp*]:
(*x* ::'*a*::*trans-reals*) − ∞ ≠ *0* ∧ (*x* ::'*a*::*trans-reals*) − ∞ ≠ ∞
by (*simp add*: *subtract-infinity*)

lemma *minus-nullD*: *x* − *y* = (*0* ::'*a*::*trans-reals*) ⟹ *x* = *y*
apply (*case-tac* y = Φ, *simp*)
apply (*case-tac* y=∞, *simp*)
apply (*case-tac* y = −∞, *simp*)
apply (*drule-tac* f=λ *a*. *a* + *y* **in** *arg-cong*)
apply (*erule-tac* Q=*?x* = *?y* **in** *contrapos-pp*)

by (*simp add*: *A6 A1*[*THEN sym*] *uminus-x-add-x*)

lemma *add-cancel-right*: ⟦ $x + a = y + a$; $a \neq \Phi$; $a \neq \infty$; $a \neq -\infty$ ⟧
$\Longrightarrow x = (y :: 'a :: trans\text{-}reals)$
apply (*drule-tac* $f = \lambda\, x.\ x - a$ **in** *arg-cong*)
by (*simp add*: *x-add-y-minus-y*)

lemma *add-cancel-left*: ⟦ $a + x = a + y$; $a \neq \Phi$; $a \neq \infty$; $a \neq -\infty$ ⟧
$\Longrightarrow x = (y :: 'a :: trans\text{-}reals)$
apply (*drule-tac* $f = \lambda\, x.\ x - a$ **in** *arg-cong*)
apply (*simp add*: *A6*)
apply (*erule-tac* $Q = ?a = ?b$ **in** *contrapos-pp*)
apply (*subst add-commute*)
apply (*subst add-commute* [**where** $a=a$])
apply (*simp add*: *add-assoc*)
by (*simp add*: *A6*[*THEN sym*] *A8*)

lemma *add-eq-infinity-iff*:
$(x + y = (\infty :: 'a :: trans\text{-}reals)) =$
$((x = \infty \ \wedge\ y \neq \Phi \wedge y \neq -\infty) \vee (y = \infty \ \wedge\ x \neq \Phi \wedge x \neq -\infty))$
apply *safe*
apply (*simp-all add*: *A6*[*THEN sym*] *A5 uminus-add-eq-minus*) **defer**
apply (*subst A2*, *simp add*: *A5*)
apply (*case-tac* $y=\Phi$, *simp*)
apply (*case-tac* $y=-\infty$, *simp add*: *A6*[*THEN sym*])
apply (*rule-tac* $a=y$ **in** *add-cancel-right*)
by (*simp add*: *A5*, *assumption*+)

lemma *add-eq-minus-infinity-iff*:
$(x + y = (-\infty :: 'a :: trans\text{-}reals)) =$
$((x = -\infty \ \wedge\ y \neq \Phi \wedge y \neq \infty) \vee (y = -\infty \ \wedge\ x \neq \Phi \wedge x \neq \infty))$
apply *safe*
apply (*simp-all add*: *A6*[*THEN sym*] *A10 uminus-add-eq-minus*) **defer**
apply (*erule contrapos-pp*, *subst A2*, *simp*)
apply (*case-tac* $y=\Phi$, *simp*)
apply (*case-tac* $y=\infty$, *simp add*: *A6*[*THEN sym*])
apply (*rule-tac* $a=y$ **in** *add-cancel-right*)
by (*simp add*: *uminus-add-eq-minus A10*, *assumption*+)

lemma *reals-add*: !! $x :: 'a :: trans\text{-}reals$.
⟦ $x \neq \Phi$; $x \neq \infty$; $x \neq -\infty$; $y \neq \Phi$; $y \neq \infty$; $y \neq -\infty$
⟧ $\Longrightarrow (x + y) \neq \Phi \wedge (x + y) \neq \infty \ \wedge (x + y) \neq -\infty$
apply *safe*
apply (*simp add*: *add-eq-nullity-iff*)
apply (*simp add*: *add-eq-infinity-iff*)
by (*simp add*: *add-eq-minus-infinity-iff*)

2.5 Closure of reals under addition

lemma *reals-add-closed*: $[\![\ x \in reals;\ y \in reals\ ]\!] \Longrightarrow x + y \in reals$

by (*unfold reals-def*, *simp add*: *reals-add*)

Distributivity of uminus over addition

lemma *uminus-distrib-add*: $-(x + y) = - x - (y :: 'a :: trans\text{-}reals)$
apply (*case-tac* $x{=}\Phi$, *simp*)
apply (*tactic* *ALLGOALS*(*case-tac* $y{=}\Phi$), *simp-all*)
apply (*case-tac* $x{=}\infty$, *simp*)
apply (*tactic* *ALLGOALS*(*case-tac* $y{=}\infty$), *simp-all*)
apply (*tactic* *ALLGOALS*(*case-tac* $x{=}-\infty$), *simp-all*)
apply (*tactic* *ALLGOALS*(*case-tac* $y{=}-\infty$))
apply (*simp-all add*: *A6*[*THEN sym*] *uminus-add-eq-minus infinity-minus A10*)

apply (*simp add*: *A5*, *subst uminus-minus-commute*)
apply (*simp add*: *A10*)
apply (*subst A2*, *simp add*: *A5*)

apply (*rule-tac* $a = x$ **in** *add-cancel-left*)
apply (*simp add*: *A6 A1*)
apply (*simp add*: *A6*[*THEN sym*] *A8*)

apply (*rule-tac* $a = y$ **in** *add-cancel-left*)
apply (*simp add*: *A6*[*THEN sym*] *A8*)
apply (*simp add*: *A6*)
apply (*subst A1*[**where** $a = y$])
apply (*simp add*: *A6*[*THEN sym*])
apply (*subst A2* [**where** $a{=}y$])
apply (*rule A8*)
by (*simp-all add*: *reals-add*)

lemma *uminus-distrib-minus*: $-(x - y) = - x + (y :: 'a :: trans\text{-}reals)$
by (*simp add*: *A6 uminus-distrib-add*)

lemma *ordering-opp*: $(x - y < 0) = (x < (y :: 'a :: trans\text{-}reals))$
apply (*subst A26* [*THEN sym*])
apply (*simp add*: *A6 uminus-distrib-add uminus-add-eq-minus*)
by (*simp add*: *A26 A6*[*THEN sym*])

Trichotomy

lemma *trichotomy*:
$[\![\ x \neq \Phi;\ y \neq (\Phi :: 'a :: trans\text{-}reals)\ ]\!] \Longrightarrow (x < y) \mid (x = y) \mid (y < x)$
apply (*cut-tac* $a{=}((x{-}y) :: 'a)$ **in** *A28*, *auto*)
apply (*simp-all add*: *A26*)
apply (*simp add*: *minus-nullity-eq-iff*, *safe*)
apply (*erule minus-nullD*)
by (*simp add*: *ordering-opp*)

lemma *reals-uminus*: !! $x::'a::trans\text{-}reals$.
$[\![\ x \neq \Phi;\ x \neq \infty;\ x \neq -\infty\ ]\!] \Longrightarrow -x\ \neq \Phi \wedge -x \neq \infty\ \wedge -x \neq -\infty$
by (*safe*, *simp-all*)

2.6 Closure of reals under uminus

lemma *reals-uminus-closed*: $x \in reals \Longrightarrow -x \in reals$
by (*unfold reals-def*, *simp add*: *reals-add*)

Closure of positives transreals under addition

lemma *zero-less-add*:
!! $a::'a::trans\text{-}reals$. $[\![\ 0 < a;\ 0 < b\ ]\!] \Longrightarrow 0 < a + b$
apply (*subst A23*[*THEN sym*])
apply (*subst A29*, *auto*)
apply (*simp add*: *sgn-positive-iff*[*THEN iffD2*])
by (*simp add*: *A23*[*THEN iffD2*])

lemma *reals-interval*: $reals = \{x.\ -\infty < x \wedge x < \infty\}$
by (*auto simp*: *reals-def less-infinity-iff minus-infinity-less-iff*)

lemma *reals-cases*: $x \in reals \Longrightarrow (x < 0) \mid x = 0 \mid (0 < x)$
apply (*unfold reals-def*, *clarsimp*)
by (*cut-tac A28* [**where** *a*=*x*], *simp*)

lemma *not-infinity-less*[*simp*]: $\neg\ (\infty < (x::'a::trans\text{-}reals))$
by (*subst A26*[*THEN sym*], *simp add*: *subtract-infinity*)

lemma *not-less-minus-infinity*[*simp*]: $\neg\ ((x::'a::trans\text{-}reals) < -\infty)$
apply (*subst A26*[*THEN sym*])
apply (*subst uminus-minus-commute*)
by (*simp add*: *subtract-infinity*)

Left distributivy of multiplication over addition

lemma *distrib-left*:
$\neg\ ((c = \infty \vee c = -\infty) \wedge sgn\ a \neq sgn\ b \wedge (a + b \notin \{0,\Phi\}))$
$\Longrightarrow\ (a+b) * c = (a * c)+(b * (c::'a::trans\text{-}reals))$

apply (*subst A13*[**where** *a*=*a*+*b*])
apply (*subst A13*[**where** *a*=*a*])
apply (*subst A13*[**where** *a*=*b*])
by (*simp add*: *A29*)

lemma *zero-mult-not-less-zero*:$\neg\ ((0::'a::trans\text{-}reals) * x < 0)$
by (*simp add*: *A24*[*THEN sym*] *A12*)

lemma *not-zero-less-zero-mult*:$\neg\ (0 < (0::'a::trans\text{-}reals) * x)$
by (*simp add*: *A23*[*THEN sym*] *A12*)

lemma *zero-mult-eq-zero-or-nullity*: $(0::'a::trans\text{-}reals) * x = 0 \vee 0 * x = \Phi$
apply (*cut-tac A28* [**where** $a=0 * x$])
by (*auto simp*: *zero-mult-not-less-zero not-zero-less-zero-mult*)

lemma *zero-mult-minus-one*: $(0::'a::trans\text{-}reals) * - 1 = 0$
apply (*cut-tac zero-mult-eq-zero-or-nullity* [**where** $x=- 1::'a$], *clarsimp*)
apply (*drule-tac* $f = \lambda x.\ x + 0 * (1 + 1)$ **in** *arg-cong*)
by (*simp add*: *A29*[*THEN sym*] *A1 uminus-add-eq-minus A8*)

lemma *zero-mult-zero*: $(0::'a::trans\text{-}reals) * 0 = 0$
apply (*subgoal-tac* $(0::'a::trans\text{-}reals) * (1 + - 1) = 0$)
apply (*simp add*: *A6* [*THEN sym*])
by (*simp add*: *A29 zero-mult-minus-one*)

Multiplicative inverse

lemma *mult-inverse*: $[\![\ a \neq 0;\ a\text{: }reals\ ]\!] \Longrightarrow a * inverse\ a = (1::'a::trans\text{-}reals)$
by (*unfold reals-def*, *clarsimp*, *drule A18*, *assumption*+, *simp add*: *A17*)

lemma *zero-mult-real-not-zero*: $[\![\ (x::'a::trans\text{-}reals)\text{: }reals;\ x \neq 0\ ]\!] \Longrightarrow 0 * x = 0$
apply (*cut-tac zero-mult-eq-zero-or-nullity* [**where** $x=x$], *safe*)
apply (*drule-tac* $f=\lambda y.\ y * inverse\ x$ **in** *arg-cong*)
by (*simp add*: *A12* [*THEN sym*] *mult-inverse*)

Multiplication with zero

lemma *zero-mult-reals*: $x : reals \Longrightarrow 0 * x = (0::'a::trans\text{-}reals) \wedge x * 0 = 0$
apply (*rule context-conjI*)
apply (*case-tac* $x=0$)
apply (*simp add*: *zero-mult-zero*)
apply (*erule zero-mult-real-not-zero*, *assumption*)
by (*subst A13*, *assumption*)

lemmas *zero-mult* = *zero-mult-reals* [*unfolded reals-def*, *simplified*]

lemma *mult-zero*: $x \neq \Phi \wedge x \neq \infty \wedge x \neq - \infty \Longrightarrow x * (0::'a::trans\text{-}reals) = 0$
apply (*rule trans*) **defer**
apply (*rule zero-mult* [*THEN conjunct1*], *assumption*)
by (*rule A13*)

Multiplication with -1

lemma *minus-one-mult-uminus-reals*:
$x : reals \Longrightarrow (- 1 ::'a::trans\text{-}reals) * x = -x$
apply (*rule-tac* $a=x$ **in** *add-cancel-right*, *unfold reals-def*, *safe*)
apply (*simp add*: *uminus-add-eq-minus A8*)
apply (*subst A14*[**where** $a=x$, *THEN sym*]) **back**
apply (*subst A13*)
apply (*subst A13*) **back**
apply (*subst A29*[*THEN sym*])
apply *simp*

by (*simp add*: *uminus-add-eq-minus A8 mult-zero*)

lemma *minus-one-mult-minus-one*[*simp*]: (− *1* ::*'a*::*trans-reals*) ∗ − *1* = *1*
by (*simp add*: *minus-one-mult-uminus-reals reals-def*)

lemma *minus-one-mult-uminus*:
(− *1* ::*'a*::*trans-reals*) ∗ *x* = −*x*
apply (*case-tac x*=Φ, *simp add*: *A13*)
apply (*case-tac x*=∞, *simp*)
apply (*case-tac x*=−∞, *clarsimp*)
apply (*subst minus-one-mult-infinity*[*THEN sym*])
apply (*subst A12*)
apply (*subst minus-one-mult-minus-one*, *simp*)
apply (*rule minus-one-mult-uminus-reals*)
by (*simp add*: *reals-def*)

lemma *minus-one-mult-minus-infinity*[*simp*]: (− *1* ::*'a*::*trans-reals*) ∗ − ∞ = ∞
by (*simp add*: *minus-one-mult-uminus*)

lemma *mult-nullity-right*[*simp*]: (*x*::*'a*::*trans-reals*) ∗ Φ = Φ
by (*subst A13*, *rule A15*)

lemma *zero-mult-minus-infinity*[*simp*]: (*0*::*'a*::*trans-reals*) ∗ − ∞ = Φ
apply (*subst minus-one-mult-infinity*[*THEN sym*])
apply (*subst A12*)
by (*simp add*: *zero-mult*)

lemma *minus-infinity-mult-zero*[*simp*]: − ∞ ∗ (*0*::*'a*::*trans-reals*) = Φ
by (*subst A13*, *rule zero-mult-minus-infinity*)

lemma *uminus-mult-left*: −((*x*::*'a*::*trans-reals*) ∗ *y*) = (− *x*) ∗ *y*
apply (*subst minus-one-mult-uminus*[*THEN sym*])
apply (*subst A12*)
by (*simp add*: *minus-one-mult-uminus*)

lemma *uminus-mult-right*: −((*x*::*'a*::*trans-reals*) ∗ *y*) = *x* ∗ (− *y*)
apply (*subst minus-one-mult-uminus*[*THEN sym*])
apply (*subst A13*)
apply (*subst A12* [*THEN sym*])
apply (*subst A13* [**where** *a*=*y*])
by (*simp add*: *minus-one-mult-uminus*)

lemma *uminus-mult-uminus* [*simp*]: −(*x*::*'a*::*trans-reals*) ∗ (− *y*) = *x* ∗ *y*
apply (*subst uminus-mult-left*[*THEN sym*])
by (*simp add*: *uminus-mult-right*)

lemma *zero-less-uminus-iff-less-zero*: (*0* < − *x*) = ((*x*::*'a*::*trans-reals*) < *0*)
apply (*simp add*: *A23*[*THEN sym*] *A24*[*THEN sym*], *auto*)
apply (*drule-tac f* =λ *x*. −*x* **in** *arg-cong*)

apply (*simp add*: *uminus-mult-right*)
apply (*drule-tac f* $=\lambda\ x.\ -x$ **in** *arg-cong*)
by (*simp add*: *uminus-mult-right*)

lemma *uminus-less-zero-iff-zero-less*: $(-x < 0) = (0 < (x::'a::trans\text{-}reals))$
apply (*simp add*: *A23*[*THEN sym*] *A24*[*THEN sym*], *auto*)
apply (*drule-tac f* $=\lambda\ x.\ -x$ **in** *arg-cong*)
apply (*simp add*: *uminus-mult-right*)
apply (*drule-tac f* $=\lambda\ x.\ -x$ **in** *arg-cong*)
by (*simp add*: *uminus-mult-right*)

lemma *less-not-sym*: $(x::'a::trans\text{-}reals) < y \Longrightarrow \neg\ (y < x)$
apply (*rule notI*, *drule A26*[*THEN iffD2*], *drule A26*[*THEN iffD2*])
apply (*drule uminus-less-zero-iff-zero-less*[*THEN iffD2*])
apply (*simp add*: *uminus-distrib-minus uminus-add-eq-minus*)
by (*erule not-less-zero-and-zero-less*, *assumption*)

Closure of positive transreals under multiplication

lemma *mult-gt-zero-gt-zero*:
!! $a::'a::trans\text{-}reals.\ [\![\ 0 < a;\ 0 < b\ ]\!] \Longrightarrow 0 < a * b$
apply (*subst A23*[*THEN sym*])
apply (*subst A12*)
apply (*subst A23*[*THEN iffD2*], *assumption*)
by (*erule A23*[*THEN iffD2*])

Mixed-sign multiplication

lemma *mult-gt-zero-less-zero*:
!! $a::'a::trans\text{-}reals.\ [\![\ 0 < a;\ b < 0\ ]\!] \Longrightarrow a * b < 0$
apply (*subst A24*[*THEN sym*])
apply (*subst A12*)
apply (*subst A23*[*THEN iffD2*], *assumption*)
by (*erule A24*[*THEN iffD2*])

lemma *mult-less-zero-gt-zero*:
!! $a::'a::trans\text{-}reals.\ [\![\ a < 0;\ 0 < b\ ]\!] \Longrightarrow a * b < 0$
by (*subst A13*, *erule mult-gt-zero-less-zero*, *assumption*)

Multiplication of two negative transreals

lemma *mult-less-zero-less-zero*:
!! $a::'a::trans\text{-}reals.\ [\![\ a < 0;\ b < 0\ ]\!] \Longrightarrow 0 < a * b$
apply (*subst A23*[*THEN sym*])
apply (*subst A12*)
apply (*subst A24*[*THEN iffD2*], *assumption*)
apply (*subst minus-one-mult-infinity* [*THEN sym*])
apply (*subst A12*[*THEN sym*])
by (*subst A24* [*THEN iffD2*], *auto*)

lemma *infinity-mult*:
$\infty * (x::'a::trans\text{-}reals) =$

(*if* $x < 0$ *then* $-\infty$ *else if* $0 < x$ *then* ∞ *else* Φ)
apply (*auto simp add*: *A23 A24*)
by (*cut-tac A28* [**where** $a{=}x$], *auto*)

lemma *mult-infinity*:
($x{::}'a{::}$*trans-reals*) $*$ ∞ =
(*if* $x < 0$ *then* $-\infty$ *else if* $0 < x$ *then* ∞ *else* Φ)
by (*subst A13*, *rule infinity-mult*)

lemma *minus-infinity-mult*:
$-\infty$ $*$ ($x{::}'a{::}$*trans-reals*) =
(*if* $x < 0$ *then* ∞ *else if* $0 < x$ *then* $-\infty$ *else* Φ)
apply (*subst minus-one-mult-infinity*[*THEN sym*])
apply (*subst A12* [*THEN sym*])
apply (*subst infinity-mult*)
by (*auto simp*: *minus-one-mult-uminus*)

lemma *mult-minus-infinity*:
($x{::}'a{::}$*trans-reals*) $*$ $-\infty{=}$
(*if* $x < 0$ *then* ∞ *else if* $0 < x$ *then* $-\infty$ *else* Φ)
by (*subst A13*, *rule minus-infinity-mult*)

Some inverse rules

lemma *inverse-infinity*[*simp*]: *inverse* (∞) = (0 ::$'a{::}$*trans-reals*)
apply (*subst A20*[*THEN sym*])
apply (*rule A19*)
by (*cut-tac* $a{=}(\infty{::}'a)$ **in** *A28*, *auto*)

lemma *inverse-nullity-iff*:
!! $x{::}'a{::}$*trans-reals*. (*inverse* x = Φ) = (x = Φ)
apply (*safe*, *simp-all*)
apply (*frule-tac* $f{=}$*inverse* **in** *arg-cong*)
apply (*case-tac* $x{=}-\infty$, *simp*)
by (*thin-tac inverse* x = Φ, *simp add*: *A19*)

lemma *inverse-noteq-zero*:
!! $x{::}'a{::}$*trans-reals*. $[\![$ $x \neq \infty$; $x \neq -\infty$ $]\!]$ $\Longrightarrow$ *inverse* $x \neq 0$
apply (*case-tac* $x{=}\Phi$, *simp*)
apply (*subst inverse-infinity*[*THEN sym*])
apply (*rule notI*)
by (*drule-tac* $f{=}$*inverse* **in** *arg-cong*, *simp add*: *A19*)

lemma *inverse-zero-iff*[*simp*]:
!! $x{::}'a{::}$*trans-reals*. (*inverse* x = 0) = (x = ∞ | x = $-\infty$)
by (*auto*, *rule classical*, *cut-tac* $x{=}x$ **in** *inverse-noteq-zero*, *auto*)

lemma *inverse-infinity-iff*[*simp*]:
!! $x{::}'a{::}$*trans-reals*. (*inverse* x = ∞) = (x = 0)
apply (*case-tac* $x{=}\Phi$, *simp*)

apply (*case-tac* $x{=}\infty$, *simp*)
apply (*case-tac* $x{=}-\infty$, *simp*)
apply (*case-tac* $x{=}0$, *auto*)
apply (*drule-tac* $f{=}\lambda\ a.\ x * a$ **in** *arg-cong*)
apply (*simp add*: *mult-inverse reals-def mult-infinity*)
apply (*case-tac* $x < 0$, *simp*)
by (*case-tac* $0 < x$, *simp-all*)

lemma *inverse-minus-infinity-iff*[*simp*]: !! $x{::}'a{::}$*trans-reals*. (*inverse* $x \neq -\infty$)
apply (*case-tac* $x{=}\Phi$, *simp*)
apply (*case-tac* $x{=}\infty$, *simp*)
apply (*case-tac* $x{=}-\infty$, *simp*)
apply (*case-tac* $x{=}0$, *auto*)
apply (*drule-tac* $f{=}\lambda\ a.\ x * a$ **in** *arg-cong*)
apply (*simp add*: *mult-inverse reals-def mult-minus-infinity*)
apply (*case-tac* $x < 0$, *simp*)
by (*case-tac* $0 < x$, *simp-all*)

2.7 Closure of positive reals under inverse

lemma *zero-less-inverse*:
!! x ::$'a{::}$*trans-reals*. $[\![\ 0 < x;\ x \neq \infty\ ]\!] \Longrightarrow 0 <$ *inverse* x
apply (*case-tac* $x{=}\Phi$, *simp*)
apply (*case-tac* $x{=}0$, *simp*)
apply (*case-tac* $x{=}-\infty$, *simp*)
apply (*cut-tac* $a{=}x$ **in** *A18*, *assumption*+)
apply (*simp add*: *A17*)
apply (*subgoal-tac inverse* $x \neq \Phi$) **defer**
apply (*simp add*: *inverse-nullity-iff*)
apply (*drule-tac trichotomy*[**where** $y = 0$]) **back**
apply (*auto*)
by (*drule-tac mult-gt-zero-less-zero*, *assumption*, *simp*)

lemma *inverse-less-zero*:
$[\![\ (x{::}'a{::}$*trans-reals*$) < 0;\ x \neq -\infty\ ]\!] \Longrightarrow$ *inverse* $x < 0$
apply (*case-tac* $x{=}\Phi$, *simp*)
apply (*case-tac* $x{=}0$, *simp*)
apply (*case-tac* $x{=}\infty$, *simp*)
apply (*cut-tac* $a{=}x$ **in** *A18*, *assumption*+)
apply (*simp add*: *A17*)
apply (*subgoal-tac inverse* $x \neq \Phi$) **defer**
apply (*simp add*: *inverse-nullity-iff*)
apply (*cut-tac reals-cases* [**where** $x =$ *inverse* x],*auto*)
apply (*drule-tac mult-less-zero-gt-zero*, *assumption*, *simp*)
by (*auto simp*: *reals-def*)

lemma *inverse-less-zero-iff*[*simp*]:
($x{::}'a{::}$*trans-reals*$) \neq -\infty \Longrightarrow$ (*inverse* $x < 0$) $= (x < 0)$
apply (*rule iffI*)

apply (*drule-tac inverse-less-zero*, *simp*)
apply (*simp add*: *A19*)
by (*erule inverse-less-zero*, *assumption*)

lemma *zero-less-inverse-iff* [*simp*]:
$[\![(x::'a::trans\text{-}reals) \neq \infty;\ x \neq 0]\!] \Longrightarrow (0 < inverse\ x) = (0 < x)$
apply (*case-tac* $x{=}-\infty$, *simp*)
apply (*rule iffI*)
apply (*drule-tac zero-less-inverse*, *simp*)
apply (*simp add*: *A19*)
by (*erule zero-less-inverse*, *assumption*)

lemma *less-not-nullity*:
$(x::'a::trans\text{-}reals) < y \Longrightarrow x \neq \Phi \wedge y \neq \Phi \wedge x \neq \infty \wedge y \neq -\infty$
by (*auto*)

2.8 Closure of reals under multiplication

lemma *reals-mult-closed*: $[\![x : reals;\ y : reals]\!] \Longrightarrow x * y : reals$
apply (*frule-tac* $x{=}x$ **in** *reals-cases*, *frule-tac* $x{=}y$ **in** *reals-cases*, *safe*)
apply (*simp-all add*: *zero-mult-reals*)

apply (*frule mult-less-zero-less-zero*, *assumption*) **back**
apply (*subst reals-def*, *clarsimp simp*: *less-not-nullity*)
apply (*drule-tac* $f = \lambda\ a.\ a * inverse\ y$ **in** *arg-cong*)
apply (*simp add*: *A12* [*THEN sym*]
mult-inverse less-zero-imp-not-zero infinity-mult reals-def)

apply (*frule mult-less-zero-gt-zero*, *assumption*)
apply (*subst reals-def*, *clarsimp simp*: *less-not-nullity*)
apply (*drule-tac* $f = \lambda\ a.\ a * inverse\ y$ **in** *arg-cong*)
apply (*frule-tac* $y{=}y$ **in** *less-not-sym*)
apply (*simp add*: *A12* [*THEN sym*]
mult-inverse zero-less-imp-not-zero minus-infinity-mult reals-def)

apply (*frule mult-gt-zero-less-zero*, *assumption*)
apply (*subst reals-def*, *clarsimp simp*: *less-not-nullity*)
apply (*drule-tac* $f = \lambda\ a.\ a * inverse\ y$ **in** *arg-cong*)
apply (*simp add*: *A12* [*THEN sym*]
mult-inverse less-zero-imp-not-zero minus-infinity-mult reals-def)

apply (*frule mult-gt-zero-gt-zero*, *assumption*) **back**
apply (*subst reals-def*, *clarsimp simp*: *less-not-nullity*)
apply (*drule-tac* $f = \lambda\ a.\ a * inverse\ y$ **in** *arg-cong*)
apply (*frule-tac* $y{=}y$ **in** *less-not-sym*)
by (*simp add*: *A12* [*THEN sym*]
mult-inverse zero-less-imp-not-zero infinity-mult reals-def)

end

3 A Model for Transarithmetic

theory *TransNumberModel*

imports *TransNumberAxclass Real*

begin

datatype *'a trans-number = P 'a | Infinity | MinusInfinity | Nullity*

lemma *inv-R-R*[*simp*]: *inv P (P x) = x*
by (*rule inv-f-f*, *unfold inj-on-def*, *auto*)

consts
primitive :: *'a trans-number ⇒ bool*
primrec
primitive (P x) = True
primitive Infinity = False
primitive MinusInfinity = False
primitive Nullity = False

lemma *primitive-iff-exists*: *primitive x = (∃ r. x = P r)*
by (*case-tac x*, *auto*)

lemma *primitiveD*: *primitive x ⟹ ∃ r. x = P r*
by (*simp add*: *primitive-iff-exists*)

instance *trans-number* :: (*zero*) *zero* **..**
instance *trans-number* ::(*type*) *infinity* **..**
instance *trans-number* ::(*type*) *nullity* **..**
instance *trans-number* :: (*plus*) *plus* **..**
instance *trans-number* :: (*minus*) *minus* **..**

defs (**overloaded**)
trans-number-zero-def: *0 == P 0*

trans-number-infinity-def: $\infty == Infinity$
trans-number-nullity-def: $\Phi == Nullity$

Note type annotations below, and overloaded symbol awkwardness

4 A model for axiomatic class trans_add

primrec
$P\ (x::'a::\{plus,minus\}) + y\ =$
$(\ if\ primitive\ y\ then\ P\ (x + inv\ P\ y)\ else$
$if\ y = \infty\ \ then\ \infty\ else$
$if\ y = -\infty\ \ then\ -\infty$
$else\ \Phi)$
$Infinity + (y::'a::\{plus,minus\}\ trans\text{-}number)$
$= (if\ primitive\ y \vee y = \infty\ then\ \infty\ else\ \Phi)$
$MinusInfinity + (y::'a::\{plus,minus\}\ trans\text{-}number)$
$= (if\ primitive\ y \vee y = -\infty\ then\ -\infty\ else\ \Phi)$
Nullity-add-left:
$Nullity + (y::'a::\{plus,minus\}\ trans\text{-}number) = \Phi$

primrec
uminus-P: $-\ P\ x = P\ (-\ (x))$
uminus-Infinity: $-\ Infinity = MinusInfinity$
uminus-MinusInfinity: $-\ MinusInfinity = Infinity$
uminus-Nullity: $-\ Nullity = Nullity$

A6

defs (**overloaded**)
trans-number-subtract-def:
$!!\ (y::'a::\{plus,minus\}\ trans\text{-}number).$
$x - y == x + (-\ y)$

lemmas *trans-number-defs* =
trans-number-zero-def trans-number-infinity-def
trans-number-nullity-def
trans-number-subtract-def

lemma *trans-number-minus-infinity-sym-def*: $MinusInfinity == -\infty$
by (*simp add*: *trans-number-defs*)

lemmas *trans-number-sym-defs* =
trans-number-zero-def[*symmetric*]
trans-number-infinity-def [*symmetric*]
trans-number-minus-infinity-sym-def
trans-number-nullity-def [*symmetric*]
trans-number-subtract-def [*symmetric*]

lemma *primitive-zero*[*simp*]: *primitive 0*
by (*unfold trans-number-zero-def*, *simp*)

lemma *not-primitive-simps*[*simp*]:
$\neg$ *primitive* $(\infty) \wedge \neg$ *primitive* $(-\infty) \wedge \neg$ *primitive* Φ
by (*unfold trans-number-defs*, *simp*)

lemma *primitive-iff*:
primitive $x = (x \neq \infty \wedge x \neq -\infty \wedge x \neq \Phi)$

by (*case-tac* x, *simp-all add*: *trans-number-defs*)

4.1 Distinctness of special values

lemma *P-neq-infinity*[*simp*]: $P\ x \neq \infty$
by (*unfold trans-number-defs*, *simp*)

lemma *P-neq-minus-infinity*[*simp*]: $P\ x \neq -\infty$
by (*unfold trans-number-defs*, *simp*)

lemma *P-neq-nullity*[*simp*]: $P\ x \neq \Phi$
by (*unfold trans-number-defs*, *simp*)

lemma *zero-neq-infinity*[*simp*]: $(0::'a::$*zero trans-number*$) \neq \infty$
by (*unfold trans-number-defs*, *simp*)

lemma *zero-neq-minus-infinity*[*simp*]:
$(0::'a::\{$*zero*,*minus* $\}$ *trans-number*$) \neq -\infty$
by (*unfold trans-number-defs*, *simp*)

lemma *zero-neq-nullity*[*simp*]: $(0::'a::\{$*zero*,*minus* $\}$ *trans-number*$) \neq \Phi$
by (*unfold trans-number-defs*, *simp*)

lemma *infinity-neq-minus-infinity*[*simp*]: $(\infty::'a::$*minus trans-number*$) \neq -\infty$
by (*unfold trans-number-defs*, *simp*)

lemma *infinity-neq-nullity*[*simp*]: $(\infty::'a::$*minus trans-number*$) \neq \Phi$
by (*unfold trans-number-defs*, *simp*)

lemma *minus-infinity-neq-nullity*[*simp*]: $(-\infty::'a::$*minus trans-number*$) \neq \Phi$
by (*unfold trans-number-defs*, *simp*)

declare *P-neq-infinity* [*THEN not-sym*,*simp*]
declare *P-neq-minus-infinity* [*THEN not-sym*,*simp*]
declare *P-neq-nullity* [*THEN not-sym*,*simp*]
declare *zero-neq-infinity* [*THEN not-sym*,*simp*]
declare *zero-neq-minus-infinity* [*THEN not-sym*,*simp*]
declare *zero-neq-nullity* [*THEN not-sym*,*simp*]
declare *infinity-neq-minus-infinity* [*THEN not-sym*,*simp*]

declare *infinity-neq-nullity* [*THEN not-sym*,*simp*]
declare *minus-infinity-neq-nullity* [*THEN not-sym*,*simp*]

lemma *P-add-P*[*simp*]: $P\ x + P\ y = P\ ((x :: 'a :: \{plus, minus\}) + y)$
by (*simp add*: *trans-number-subtract-def*)

lemma *P-add-non-primitive*[*simp*]: $P\ x + \infty = \infty \wedge P\ x - \infty = -\infty \wedge P\ x + \Phi = \Phi$
by (*simp add*: *trans-number-subtract-def*)

lemma *Infinity-add-left*[*simp*]:
$\infty + P\ x = (\infty :: 'a :: \{plus, minus\}\ trans\text{-}number) \wedge$
$\infty + \infty = (\infty :: 'a\ trans\text{-}number) \wedge$
$\infty - \infty = (\Phi :: 'a\ trans\text{-}number) \wedge$
$\infty + \Phi = (\Phi :: 'a\ trans\text{-}number)$

by (*simp add*: *trans-number-defs*)

lemma *MinusInfinity-add-left*[*simp*]:
$-\infty + P\ x = (-\infty :: 'a :: \{plus, minus\}\ trans\text{-}number) \wedge$
$-\infty + \infty = (\Phi :: 'a\ trans\text{-}number) \wedge$
$-\infty - \infty = (-\infty :: 'a\ trans\text{-}number) \wedge$
$-\infty + \Phi = (\Phi :: 'a\ trans\text{-}number)$

by (*simp add*: *trans-number-defs*)

lemma *uminus-simps*[*simp*]:
$-\ P\ x = (P\ (-\ x) :: 'a :: \{minus\}\ trans\text{-}number) \wedge$
$-\ (-\infty) = (\infty :: 'a\ trans\text{-}number) \wedge$
$-\ \Phi = (\Phi :: 'a\ trans\text{-}number)$

by (*simp add*: *trans-number-defs*)

A4

lemma *nullity-add-left*[*simp*]:
$\Phi + x = (\Phi :: 'a :: \{plus, minus\}\ trans\text{-}number)$

by (*simp add*: *trans-number-nullity-def*)

lemma *nullity-subtract-left*[*simp*]:
$\Phi - x = (\Phi :: 'a :: \{plus, minus\}\ trans\text{-}number)$

by (*simp add*: *trans-number-defs*)

Axiom A5

lemma *addition-infinity-not-null*:
$[\![\ x \neq -\infty\ ;\ x \neq \Phi\ ]\!] \Longrightarrow (x :: 'a :: \{plus, minus\}\ trans\text{-}number) + \infty = \infty$
by (*case-tac x*, *simp-all add*: *trans-number-defs*)

A1

lemma *add-assoc*:
((x::'a::*ab-group-add trans-number*) + y) + z = x + (y + z)

apply (*case-tac* x, *safe*)
apply (*tactic ALLGOALS* (*case-tac* y), *safe*)
apply (*tactic ALLGOALS* (*case-tac* z), *safe*)
by (*simp-all add*: *trans-number-sym-defs*)

A2

lemma *add-commute*:
(x::'a::*ab-group-add trans-number*) + y = y + x

apply (*case-tac* x, *safe*)
apply (*tactic ALLGOALS* (*case-tac* y))
by (*simp-all add*: *trans-number-sym-defs*)

lemma *add-left-commute*:
a + (b + c) = b + (a + (c::'a::*ab-group-add trans-number*))
by (*rule mk-left-commute* [*of op* +, *OF add-assoc add-commute*])

theorems *trans-add-ac* = *add-assoc add-commute add-left-commute*

lemma *nullity-add-right*[*simp*]:
x + Φ = (Φ ::'a::*ab-group-add trans-number*)
by (*subst add-commute*, *simp add*: *trans-number-nullity-def*)

A3

lemma *add-identity*[*simp*]:
0 + (x::'a::*ab-group-add trans-number*) = x

apply (*unfold trans-number-zero-def*, *case-tac* x)
by (*simp-all*, *simp-all add*: *trans-number-defs*)

A7

lemma *bijectivity-of-uminus*[*simp*]:
−(−(x::'a::*ab-group-add trans-number*)) = x

by (*case-tac* x, *simp-all add*: *trans-number-sym-defs*)

A8

lemma *additive-inverse*[*simp*]:
!! (x::'a::*ab-group-add trans-number*).
primitive x ⟹ x − x = *0*

apply (*case-tac* x)
by (*simp-all add*: *trans-number-subtract-def trans-number-zero-def*)

A9

lemma *uminus-nullity*:
  $-$ (Φ ::('*a*::*minus*) *trans-number*) = Φ

  by *simp*

A10

lemma *subtraction-infinity-not-null*:
  !! (*x*::'*a*::{*plus*,*minus*} *trans-number*).
      ⟦ *x* ≠ ∞; *x* ≠ Φ ⟧ ⟹ *x* − ∞ = −∞

  by (*case-tac x*, *simp-all add*: *trans-number-defs*)

instance *trans-number* :: (*ab-group-add*) *trans-add*
  apply (*intro-classes*, *simp-all add*:)
  apply (*rule add-assoc* [*THEN sym*])
  apply (*rule add-commute*)
  apply (*subst add-commute*, *erule addition-infinity-not-null*, *assumption*)
   apply (*simp add*: *trans-number-subtract-def*)
  apply (*rule additive-inverse*, *simp add*: *primitive-iff*)
  by (*erule subtraction-infinity-not-null*, *assumption*)

5 Transnumber ordering

instance *trans-number* :: (*ord*) *ord* **..**
primrec
    P-less: *P x* < *y* = (*primitive y* ∧ (*x*::'*a*::{*minus*,*ord*}) < *inv P y* | *y* = ∞)
    Infinity-less: (*Infinity* < (*y*::'*a*::{*minus*,*ord*} *trans-number*)) = *False*
    MinusInfinity-less:
    (*MinusInfinity* < *y*) = (*y* ≠ −∞ ∧ *y* ≠ (Φ::'*a*::{*minus*,*ord*} *trans-number*))
    Nullity-less: (*Nullity* < (*y*::'*a*::{*minus*,*ord*} *trans-number*)) = *False*

defs (**overloaded**)
   trans-number-le-def:
     (*x*::'*a*::{*minus*,*ord*} *trans-number*) <= *y* == (*x* < *y* | *x* = *y*)

lemma *P-less-P*[*simp*]: (*P x* < *P y*) = (*x* < (*y*::'*a*::{*minus*,*order*}))
  by (*simp add*: *trans-number-defs*)

lemma *P-less-non-primitive*[*simp*]: (*P x* < ∞) ∧ ¬ (*P x* < −∞) ∧ ¬ (*P x* < Φ)
  by (*simp add*: *trans-number-defs*)

lemma *P-le*[*simp*]:
  (*P x* ≤ *P y* = (*x* ≤ (*y*::'*a*::{*order*,*minus*}))) ∧
     (*P x* ≤ ∞) ∧
     ¬ (*P x* ≤ −∞) ∧
     ¬ (*P x* ≤ Φ)
  by (*simp add*: *trans-number-defs trans-number-le-def order-le-less*)

lemma *not-infinity-less*[*simp*]:

$\neg$ ((∞::'a::{*minus*,*ord*} *trans-number*) < *x*)
by (*unfold trans-number-defs*,*simp*)

lemma *infinity-le* [*simp*]:
((∞::'a::{*minus*,*ord*} *trans-number*) $\leq$ *x*) = (*x* = ∞)
by (*auto simp*: *trans-number-le-def*)

lemma *le-infinity*[*simp*]: *x* $\leq$ (∞::'a::{*minus*,*ord*} *trans-number*) = (*x* $\neq$ Φ)
by (*case-tac x*, *auto simp*: *trans-number-le-def trans-number-defs*)

lemma *minus-infinity-less*[*simp*]:
($-$(∞::'a::{*minus*,*ord*} *trans-number*) < *x*) = (*x* $\neq$ $-\infty$ $\wedge$ *x* $\neq$ Φ)
by (*simp add*: *trans-number-defs*)

lemma *minus-infinity-le*[*simp*]:
($-$(∞::'a::{*minus*,*ord*} *trans-number*) $\leq$ *x*) = (*x* $\neq$ Φ)
by (*auto simp*: *trans-number-le-def*)

lemma *not-nullity-less*[*simp*]: $\neg$ ((Φ::'a::{*minus*,*ord*} *trans-number*) < *x*)
by (*unfold trans-number-nullity-def*, *simp*)

lemma *nullity-le*[*simp*]: ((Φ::'a::{*minus*,*ord*} *trans-number*) $\leq$ *x*) = (*x* = Φ)
by (*auto simp*: *trans-number-le-def*)

lemma *not-less-nullity*[*simp*]: $\neg$ (*x* < (Φ::'a::{*minus*,*ord*} *trans-number*))
by (*case-tac x*, *simp-all*)

lemma *le-nullity*[*simp*]:
(*x* $\leq$ (Φ::'a::{*minus*,*order*} *trans-number*)) = (*x* = Φ)
by (*case-tac x*, *simp-all add*: *trans-number-sym-defs*)

lemma *zero-less-P*[*simp*]:
((*0*::'a::{*minus*,*ord*,*zero*} *trans-number*) < *P x*) = (*0* < *x*)
by (*simp add*: *trans-number-zero-def*)

A25

lemma *zero-less-infinity*[*simp*]: ((*0*::'a::{*minus*,*ord*,*zero*} *trans-number*) < ∞)
by (*simp add*: *trans-number-zero-def*)

lemma *zero-not-less-minus-infinity*[*simp*]:
$\neg$ ((*0*::'a::{*minus*,*ord*,*zero*} *trans-number*) < $-\infty$)
by (*simp add*: *trans-number-zero-def*)

lemma *irreflexive-less*[*simp*]: $\neg$ ((*x*::'a::{*minus*,*order*,*zero*} *trans-number*) < *x*)
by (*case-tac x*, *auto simp*: *trans-number-sym-defs*)

declare *P-less*[*simp del*]

A26

lemma *trans-number-ordering*:
$(x - y > 0) = (x > (y::'a::lordered\text{-}ab\text{-}group\ trans\text{-}number))$
apply (*case-tac x*)
apply (*tactic ALLGOALS* (*case-tac y*))
apply (*simp-all add*: *trans-number-subtract-def*)
apply (*unfold trans-number-sym-defs*)
by (*simp-all add*: *compare-rls*)

A27 holds trivially since gt is simply a syntactic abbreviation

lemma *less-than-gt-than-eq*: $(x < y) = (y > x)$
by (*rule refl*)

A28

lemma *quadrachotomoy*:
$!!x::'a::\{ab\text{-}group\text{-}add,linorder\}\ trans\text{-}number.$
$xor\ [x < 0,\ x = 0,\ 0 < x,\ x= \Phi]$

apply (*case-tac x*)
apply (*simp-all add*: *trans-number-sym-defs*)
apply (*rule-tac x=a* **and** *y =0* **in** *linorder-cases*)
by (*auto simp*: *trans-number-zero-def*)

A29

lemma *pos-closure-add*:
$!!\ x::'a::\{lordered\text{-}ab\text{-}group\}\ trans\text{-}number.$
$[\![\ 0 < x\ ;\ 0 < y\ ]\!] \Longrightarrow 0 < x + y$

apply (*case-tac x*)
apply (*tactic ALLGOALS* (*case-tac y*))
apply (*simp-all add*: *trans-number-sym-defs*)
by (*erule add-pos-pos*, *assumption*)

lemma *trans-number-order-refl*:
$(x::'a::\{minus,order\}\ trans\text{-}number) \leq x$

apply (*case-tac x*)
by (*simp-all add*: *trans-number-sym-defs*)

lemma *trans-number-order-trans*:
$[\![\ (x::'a::\{minus,order\}\ trans\text{-}number) \leq y;\ y \leq z\ ]\!] \Longrightarrow x \leq z$

apply (*case-tac x*)
apply (*tactic ALLGOALS* (*case-tac y*))
apply (*tactic ALLGOALS* (*case-tac z*))
by (*simp-all add*: *trans-number-sym-defs*)

lemma *trans-number-order-antisym*:
$[\![\ (x::'a::\{minus,order\}\ trans\text{-}number) \leq y;\ y \leq x\ ]\!] \Longrightarrow x = y$

apply (*case-tac x*)
apply (*tactic ALLGOALS* (*case-tac y*))
by (*simp-all add*: *trans-number-sym-defs*)

lemma *trans-number-less-le*:
((*x*::'*a*::{*minus*,*order*} *trans-number*) < *y*) = (*x* ≤ *y* ∧ *x* ≠ *y*)

apply (*unfold trans-number-le-def*, *auto*)
by (*case-tac y*, *simp-all add*: *trans-number-sym-defs*)

axclass *minus-order* ⊆ *order*, *minus*

instance *pordered-ab-group-add* ⊆ *minus-order* **..**

instance *trans-number* :: (*minus-order*)*order*
apply (*intro-classes*)
apply (*rule trans-number-order-refl*)
apply (*erule trans-number-order-trans*, *assumption*)
apply (*erule trans-number-order-antisym*, *assumption*)
by (*rule trans-number-less-le*)

lemma *ext-number-linear*:
⟦ (*x* ::'*a*::{*minus*,*linorder*} *trans-number*) ≠ Φ; *y* ≠ Φ ⟧ ⟹ *x* ≤ *y* | *y* ≤ *x*
apply (*case-tac x*)
apply (*tactic ALLGOALS* (*case-tac y*))
by (*simp-all add*: *trans-number-sym-defs linorder-linear*)

lemma *not-elem-conv*: *xs* ⊆ {*x*. *x* ≠ *a*} = (*a* ∉ *xs*)
by (*unfold subset-def*, *auto*)

5.1 Lattice-completeness of trans_numbers

NOTE: cannot express l.-c. of transnumbers as instance rule

lemma *ext-number-complete-lattice*:
lattice-complete {*x* :: *real trans-number*. *x* ≠ Φ}

apply (*unfold lattice-complete-def*)
apply (*clarsimp simp*: *not-elem-conv*)
apply (*case-tac* ∃ *a*. ∀ *x* ∈ *ys*. *x* < *P a*)
apply (*case-tac* ∃ *b*. *P b* ∈ *ys*, *safe*)
apply (*cut-tac S* = *inv P* ' (*ys* − {−∞}) **in** *reals-complete*)
apply (*fastsimp*)
apply (*rule-tac x*=*a* **in** *exI*)
apply (*unfold isUb-def setle-def isLub-def leastP-def setge-def*, *safe*)
apply (*case-tac x*, *unfold trans-number-sym-defs*)
apply (*fastsimp*, *fastsimp*, *fastsimp*, *fastsimp*, *fast*)
apply (*clarsimp simp*: *isUb-def setle-def*)
apply (*rule-tac x*=*P t* **in** *exI*, *safe*, *simp-all*)
apply (*case-tac y*, *unfold trans-number-sym-defs*, *simp*)

apply (*erule-tac* $x = P\ aa$ **in** *ballE*) **back**
apply (*simp*+, *force*, *fastsimp*, *fast*)
apply (*case-tac* v, *unfold trans-number-sym-defs*)
apply *clarsimp*
apply (*erule-tac* $x = aa$ **in** *allE*, *clarsimp*)
apply (*case-tac* y, *unfold trans-number-sym-defs*)
apply (*fastsimp*, *fastsimp*, *assumption*, *fastsimp*, *simp*)
apply (*fastsimp*, *fastsimp*)
apply (*rule-tac* $x{=}-\infty$ **in** *exI*, *clarsimp*)
apply (*case-tac* y, *unfold trans-number-sym-defs*)
apply (*fastsimp*, *fastsimp*, *simp*, *fastsimp*)
apply (*rule-tac* $x = \infty$ **in** *exI*, *auto*)
apply (*case-tac* v, *unfold trans-number-sym-defs*)
apply (*auto*)
apply (*erule-tac* $x = a + 1$ **in** *allE*, *clarify*)
apply (*erule-tac* $x = x$ **in** *ballE*)
apply (*auto simp*: *trans-number-less-le*)
apply (*erule-tac notE*, *erule-tac trans-number-order-trans*, *simp*)
apply (*erule-tac* $x = 1$ **in** *allE*, *auto*)
apply (*erule-tac* $x = x$ **in** *ballE*)
by (*erule-tac notE*, *erule-tac trans-number-order-trans*, *fastsimp*+)

6 A model for axiomatic class trans_mult

instance *trans-number* :: (*one*) *one* **..**
instance *trans-number* :: (*inverse*) *inverse* **..**
instance *trans-number* :: (*times*)*times* **..**

defs (**overloaded**)
trans-number-one-def: $1 == P\ 1$

lemmas *trans-number-defs* =
trans-number-zero-def *trans-number-one-def*
trans-number-infinity-def trans-number-nullity-def
trans-number-subtract-def

lemmas *trans-number-sym-defs* =
trans-number-zero-def[*symmetric*]
trans-number-one-def[*symmetric*]
trans-number-infinity-def [*symmetric*]
trans-number-minus-infinity-sym-def
trans-number-nullity-def [*symmetric*]
trans-number-subtract-def [*symmetric*]

lemma *primitive-one*[*simp*]: *primitive 1*
by (*unfold trans-number-one-def*, *simp*)

Warning: simpsets contain different mult laws with special cases for RHS

primrec

trans-mult-P:
$P\ (x::'a::\{zero,times,minus,ord\}) * y =$
$(\ if\ primitive\ y\ then\ P\ (x * inv\ P\ y)\ else$
$if\ (y = \infty \wedge x > 0) \mid (y = -\infty \wedge x < 0)\ then\ \infty\ else$
$if\ (y = \infty \wedge x < 0) \mid (y = -\infty \wedge x > 0)\ then\ -\infty$
$else\ \Phi)$

trans-mult-Infinity:
$Infinity * (y::'a::\{zero,times,minus,ord\}\ trans\text{-}number) =$
$(if\ (primitive\ y \wedge y > 0) \mid y = \infty\ then\ \infty\ \ else$
$if\ (primitive\ y \wedge y < 0) \mid y = -\infty\ then\ -\infty$
$else\ \Phi)$

trans-mult-MinusInfinity:
$MinusInfinity * (y::'a::\{zero,times,minus,ord\}\ trans\text{-}number) =$
$(if\ (primitive\ y \wedge y < 0) \mid y = -\infty\ then\ \infty\ \ else$
$if\ (primitive\ y \wedge y > 0) \mid y = \infty\ then\ -\infty$
$else\ \Phi)$

trans-mult-Nullity: $Nullity * (y::'a::\{zero,times,minus,ord\}\ trans\text{-}number) = \Phi$

lemma *P-mult-P*[*simp*]:
$P\ x * P\ y = P\ ((x::'a::\{zero,times,minus,linorder\}) * y)$

by *auto*

A15 is first conjunct of mult_nullity, see also trans_mult_nullity

lemma *mult-nullity*[*simp*]:
$(\Phi::'a::ordered\text{-}idom\ trans\text{-}number) * x = \Phi \wedge x * \Phi = \Phi$

apply (*rule conjI*)
apply (*simp add*: *trans-number-nullity-def*)
by (*case-tac x*, *auto*)

A16 is fourth conjunct of mult_zero

lemma *mult-zero*[*simp*]:
$0 * P\ (x::'a::ordered\text{-}idom) = 0 \wedge$
$P\ x * 0 = 0 \wedge$
$(0::'a\ trans\text{-}number) * \infty = \Phi \wedge$
$\infty * (0::'a\ trans\text{-}number) = \Phi \wedge$
$(0::'a\ trans\text{-}number) * -\infty = \Phi \wedge$
$-\infty * (0::'a\ trans\text{-}number) = \Phi \wedge$
$\infty * \Phi = (\Phi::'a\ trans\text{-}number)$

by (*simp add*: *trans-number-defs*)

lemma *mult-infinity*[*simp*]:
$\infty * (\infty::'a::ordered\text{-}idom\ trans\text{-}number) = \infty \wedge$
$(\infty::'a\ trans\text{-}number)\ * -\infty = -\infty \wedge$

$-\infty * (\infty::'a\ trans\text{-}number) = -\infty \wedge$
$-\infty * -\infty = (\infty::'a\ trans\text{-}number)$

by (*simp add*: *trans-number-defs*)

lemma *P-mult-infinity-less-zero*:
!! x ::($'a$::*ordered-idom*).
$x < 0 \Longrightarrow P\ x * \infty\ = -\infty\ \wedge\ \infty * P\ x\ = -\infty$

by (*simp add*: *trans-number-defs*)

lemma *P-mult-infinity-gt-zero*:
!! x ::($'a$::*ordered-idom*).
$0 < x \Longrightarrow P\ x * \infty\ = \infty\ \wedge\ \infty * P\ x\ = \infty$

by (*simp add*: *trans-number-defs*)

lemma *P-mult-MinusInfinity-less-zero*:
!! x ::($'a$::*ordered-idom*).
$x < 0 \Longrightarrow P\ x * -\infty\ =\ \infty\ \wedge\ -\infty * P\ x\ =\ \infty$

by (*simp add*: *trans-number-defs*)

lemma *P-mult-MinusInfinity-gt-zero*:
!! x ::($'a$::*ordered-idom*).
$0 < x \Longrightarrow P\ x * -\infty\ = -\infty\ \wedge\ -\infty * P\ x\ = -\infty$

by (*simp add*: *trans-number-defs*)

lemmas *P-mult-infinities* =
P-mult-infinity-less-zero P-mult-infinity-gt-zero
P-mult-MinusInfinity-less-zero P-mult-MinusInfinity-gt-zero

declare *trans-mult-P* [*simp del*]
trans-mult-Infinity [*simp del*]
trans-mult-MinusInfinity [*simp del*]
trans-mult-Nullity [*simp del*]

A13

lemma *mult-commute*:
(x::$'a$::*ordered-idom trans-number*) $* y = y * x$

apply (*case-tac x*, *safe*)
apply (*tactic ALLGOALS* (*case-tac y*), *safe*)
apply (*simp-all add*: *trans-number-sym-defs mult-ac*)
by ((*rule-tac x=a* **and** *y =0* **in** *linorder-cases*,
(*simp add*: *trans-number-zero-def* [*symmetric*] *P-mult-infinities*)+)+)

A12

lemma *mult-assoc*:
((x::'a::*ordered-idom trans-number*) $*$ y) $*$ z $=$ x $*$ (y $*$ z)

apply (*case-tac* x, *safe*)
apply (*tactic ALLGOALS* (*case-tac* y), *safe*)
apply (*tactic ALLGOALS* (*case-tac* z), *safe*)
apply (*simp-all add*: *trans-number-sym-defs mult-ac*)
apply (*tactic ALLGOALS* (*case-tac* $a = 0$))
apply (*tactic ALLGOALS* (*case-tac* $a < 0$))
apply (*simp-all add*: *trans-number-sym-defs*
P-mult-infinities mult-ac linorder-not-less order-le-less)
apply (*tactic ALLGOALS*(*case-tac* $aa = 0$))
apply (*tactic ALLGOALS*(*case-tac* $aa < 0$))
by (*simp-all add*: *trans-number-sym-defs*
P-mult-infinities mult-ac linorder-not-less order-le-less
zero-less-mult-iff mult-less-0-iff)

lemma *mult-left-commute*:
x $*$ (y $*$ z) $=$ y $*$ (x $*$ (z::'a:: *ordered-idom trans-number*))
by (*rule mk-left-commute* [*of op* $*$, *OF mult-assoc mult-commute*])

lemmas *trans-mult-ac* $=$ *mult-assoc mult-commute mult-left-commute*

A14

lemma *mult-one-left*[*simp*]: 1 $*$ x $=$ (x::'a:: *ordered-idom trans-number*)

apply (*case-tac* x)
apply (*simp-all add*: *trans-number-one-def*)
apply (*simp add*: *P-mult-infinities trans-number-infinity-def*[*symmetric*])
apply (*simp add*: *P-mult-infinities*
uminus-Infinity [*THEN sym*] *trans-number-infinity-def*[*symmetric*])
by (*simp add*: *trans-number-nullity-def* [*symmetric*])

lemma *mult-one-right*[*simp*]: x $*$ 1 $=$ (x::'a:: *ordered-idom trans-number*)
by (*subst mult-commute*, *rule mult-one-left*)

lemma *not-primitive-mult-infinity*[*simp*]:
¬ (*primitive* (*P* (x::'a::*ordered-idom*) $*$ ∞))
apply (*case-tac* $x < 0$)
apply (*simp add*: *P-mult-infinities*)
by (*auto simp*: *linorder-not-less order-le-less*
P-mult-infinities trans-number-sym-defs)

lemma *not-primitive-mult-MinusInfinity*[*simp*]:
¬ (*primitive* (*P* (x::'a::*ordered-idom*) $*$ $-\infty$))
apply (*case-tac* $x < 0$)
apply (*simp add*: *P-mult-infinities*)
by (*auto simp*: *linorder-not-less order-le-less*
P-mult-infinities trans-number-sym-defs)

6.1 Inverse and division

primrec
inverse (*P* (*x*::'*a*::{*inverse*,*zero*})) = (*if x = 0 then* ∞ *else P* (*inverse x*))
inverse (*Infinity*::('*a*::{*inverse*,*zero*}) *trans-number*) = *0*
inverse (*MinusInfinity*::('*a*::{*inverse*,*zero*}) *trans-number*) = *0*
inverse (*Nullity*::('*a*::{*inverse*,*zero*}) *trans-number*)= Φ

A17

defs (**overloaded**)
trans-number-divison-def:
x / (*y*::'*a*::{*times*,*inverse*} *trans-number*) == *x* ∗ *inverse y*

lemma *inverse-Infinity*[*simp*]:
inverse (∞ ::('*a*::{*inverse*,*zero*}) *trans-number*) = *0*

by (*simp add*: *trans-number-defs*)

lemma *inverse-MinusInfinity*[*simp*]:
inverse (−∞ ::('*a*::{*inverse*,*zero*,*minus*}) *trans-number*) = *0*

by (*simp add*: *trans-number-defs*)

lemma *inverse-nullity*[*simp*]:
inverse (Φ ::('*a*::{*inverse*,*zero*}) *trans-number*)= Φ

by (*simp add*: *trans-number-defs*)

A18

lemma *multiplicative-inverse*:
⟦ *primitive* (*x*::'*a*::*ordered-field trans-number*); *x* ≠ *0* ⟧ ⟹ *x* / *x* = *1*

apply (*case-tac x*, *simp-all*)
by (*auto simp*: *trans-number-divison-def trans-number-sym-defs*)

A19

lemma *bij-inverse*:
(*x*::'*a*::*ordered-field trans-number*) ≠ − ∞ ⟹ *inverse* (*inverse x*) = *x*

apply (*case-tac x*)
apply (*simp-all add*: *trans-number-defs nonzero-inverse-inverse-eq*)
by (*fast intro*: *inverse-zero-imp-zero*)

A20

lemma *inverse-zero*[*simp*]:
inverse (*0*::'*a*::*ordered-field trans-number*) = ∞

by (*simp add*: *trans-number-defs*)

A21

lemma *inverse-MinusInfinity*[*simp*]:
inverse $(-\infty ::'a::$*ordered-field trans-number*$) = 0$

by (*simp add*: *trans-number-defs*)

A22

lemma *inverse-nullity*[*simp*]:
inverse $(\Phi::'a::$*ordered-field trans-number*$) = \Phi$

by (*simp add*: *trans-number-defs*)

A23

lemma *positive-inf-mult*:
$(\infty * x = \infty) = (0 <$ $(x::'a::$*ordered-field trans-number*$))$

apply (*case-tac x*)
apply (*simp-all add*: *trans-number-sym-defs*)
apply(*rule-tac x=a* **and** *y =0* **in** *linorder-cases*)
apply (*simp-all add*: *P-mult-infinities*)
by (*simp add*: *trans-number-sym-defs*)

A24

lemma *negative-inf-mult*:
$(\infty * x = -\infty) = (x <$ $(0::'a::$*ordered-field trans-number*$))$

apply (*case-tac x*)
apply (*simp-all add*: *trans-number-sym-defs*)
apply (*simp add*: *trans-number-zero-def*)
apply(*rule-tac x=a* **and** *y =0* **in** *linorder-cases*)
apply (*simp-all add*: *P-mult-infinities*)
by (*simp add*: *trans-number-sym-defs*)

instance *trans-number* :: (*ordered-idom*) *sgn* **..**

defs (**overloaded**)
trans-number-sgn-def:
sgn $(a::'a::$*ordered-idom trans-number*$)$
$==$ (*if 0 < a then 1 else*
if 0 = a then 0 else
if a < 0 then − 1 else
$(* a = \Phi *)$ Φ)

instance *trans-number* :: (*ordered-idom*) *trans-sgn*
by (*intro-classes*, *unfold trans-number-sgn-def*, *rule refl*)

lemma *P-mult-infinity-neq-infinity-iff*:
$(P\ a * \infty \neq \infty) = (a \leq (0::'a::$*ordered-idom*$))$

apply (*case-tac a< 0*, *simp add*: *P-mult-infinity-less-zero*)
by (*auto simp*: *P-mult-infinity-gt-zero*
linorder-not-less order-le-less trans-number-sym-defs)

lemma *P-mult-infinity-neq-MinusInfinity-iff*:
(*P a* $*$ $\infty \neq -\infty$) = (*0* $\leq$ (*a*::*'a*::*ordered-idom*))
apply (*case-tac a< 0*, *simp add*: *P-mult-infinity-less-zero*)
by (*auto simp*: *P-mult-infinity-gt-zero*
linorder-not-less order-le-less trans-number-sym-defs)

lemma *P-mult-MinusInfinity-neq-MinusInfinity-iff*:
(*P a* $*$ $-\infty \neq -\infty$) = (*a* $\leq$ (*0*::*'a*::*ordered-idom*))
apply (*case-tac 0 < a*, *simp add*: *P-mult-MinusInfinity-gt-zero*)
by (*auto simp*: *P-mult-MinusInfinity-less-zero*
linorder-not-less order-le-less trans-number-sym-defs)

lemma *P-mult-MinusInfinity-neq-infinity-iff*:
(*P a* $*$ $-\infty \neq \infty$) = (*0* $\leq$ (*a*::*'a*::*ordered-idom*))
apply (*case-tac 0< a*, *simp add*: *P-mult-MinusInfinity-gt-zero*)
by (*auto simp*: *P-mult-MinusInfinity-less-zero*
linorder-not-less order-le-less trans-number-sym-defs)

lemma *sgn-P*:
sgn (*P* (*x*::*'a*::*ordered-idom*)) = (*if x < 0 then − 1 else if x = 0 then 0 else 1*)
apply (*unfold trans-number-sgn-def*, *simp add*: *trans-number-sym-defs*)
by (*safe*, *simp-all add*: *trans-number-zero-def*)

lemma *uminus-eq-iff*:
(*−x* = (*x*::*'a*::*ordered-idom*)) = (*x* = *0*)
by *auto*

lemma *sgn-P-eq-iff*:
!! (*x*::*'a*::*ordered-idom*) (*y*::*'a*).
(*sgn* (*P x*) = *sgn* (*P y*))
= (((*x* < *0*) = (*y* < *0*)) ∧ ((*x* = *0*) = (*y* = *0*)) ∧
((*0* < *x*) = (*0* < (*y*::*'a*::*ordered-idom*))))
apply (*simp add*: *sgn-P*, *safe*, *simp-all*)
apply (*simp add*: *trans-number-defs*)
apply (*simp add*: *trans-number-defs*)
apply (*simp add*: *trans-number-defs*)
apply (*unfold trans-number-defs*)
apply (*simp only*: *uminus-P trans-number.simps uminus-eq-iff*)
apply (*simp*)
by (*simp only*: *uminus-P trans-number.simps uminus-eq-iff*)

lemma *sgn-zero*[*simp*]: *sgn* (*0*::(*'a*::*ordered-idom trans-number*)) = *0*
by (*simp add*: *trans-number-sgn-def*)

lemma *sgn-infinity*[*simp*]: *sgn* (∞ ::('*a*::*ordered-idom trans-number*)) = *1*
by (*simp add*: *trans-number-sgn-def*)

lemma *sgn-minus-infinity*[*simp*]: *sgn* ($-\infty$ ::('*a*::*ordered-idom trans-number*)) = *− 1*
by (*simp add*: *trans-number-sgn-def*)

lemma *sgn-zero-iff*[*simp*]:
(*sgn* (*x*::('*a*::*ordered-idom trans-number*)) = *0*) = (*x* = *0*)
by (*auto simp*: *trans-number-sgn-def*, *simp-all add*: *trans-number-defs*)

lemma *sgn-P-one-iff*[*simp*]:
(*sgn* (*P* (*x*::'*a*::*ordered-idom*)) = *1*) = (*0* < *x*)
apply (*simp add*: *trans-number-sgn-def*, *safe*, *simp*)
apply(*unfold trans-number-defs*)
apply (*simp only*: *uminus-P trans-number.simps uminus-eq-iff*)
by *simp-all*

lemma *sgn-P-minus-one-iff*[*simp*]:
(*sgn* (*P* (*x*::'*a*::*ordered-idom*)) = *− 1*) = (*x* < *0*)
apply (*simp add*: *trans-number-sgn-def*, *safe*, *simp*)
apply(*unfold trans-number-defs*)
apply (*fast*, *simp*, *simp*)
apply (*simp only*: *uminus-P trans-number.simps uminus-eq-iff*)
apply (*blast intro*: *order-less-asym*, *simp*)
apply (*simp only*: *uminus-P trans-number.simps uminus-eq-iff*)
apply (*blast intro*: *order-less-asym*, *simp*, *simp*)
apply (*simp only*: *uminus-P trans-number.simps uminus-eq-iff*)
apply (*blast intro*: *order-less-asym*)
by (*simp*, *simp*)

lemma *P-eq-zero*: (*P x* = *0*) = (*x* = *0*)
by (*unfold trans-number-zero-def*, *auto*)

A29

lemma *distributivity*:
!! *a*::('*a*::*ordered-field trans-number*).
¬ ((*a* = ∞ ∨ *a* = − ∞) ∧ *sgn b* ≠ *sgn c* ∧ (*b* + *c* ∉ {*0*,Φ}))
⟹ *a* ∗ (*b*+*c*) = (*a* ∗ *b*)+(*a* ∗ *c*)
apply (*case-tac a*, *tactic ALLGOALS Clarsimp-tac*)
apply (*tactic ALLGOALS* (*case-tac b*) *THEN ALLGOALS Clarsimp-tac*)
apply (*tactic ALLGOALS* (*case-tac c*) *THEN ALLGOALS Clarsimp-tac*)
apply (*simp-all add*: *trans-number-sym-defs*)
apply (*rule right-distrib*, *safe*)
apply (*simp add*: *P-mult-infinities trans-number-sym-defs*
P-mult-infinity-neq-infinity-iff P-mult-infinity-neq-MinusInfinity-iff)
apply (*simp add*: *P-mult-infinities trans-number-sym-defs*
P-mult-MinusInfinity-neq-infinity-iff P-mult-MinusInfinity-neq-MinusInfinity-iff)

apply (*tactic ALLGOALS* (*case-tac aa* < *0*))
apply (*simp-all add*: *linorder-not-less order-le-less*, *safe*)
apply (*simp-all add*: *trans-number-sym-defs*
P-mult-infinities sgn-P-eq-iff P-eq-zero)
apply *auto*
apply (*subst P-mult-infinity-less-zero* [*THEN conjunct2*], *simp*, *rule refl*)
apply (*subst P-mult-infinity-gt-zero* [*THEN conjunct2*], *simp*, *rule refl*)
apply (*drule sym*) **prefer** *4* **apply** (*drule sym*)
by (*simp-all add*: *P-mult-infinities trans-number-sym-defs*)

instance *trans-number* :: (*ordered-field*) *trans-mult*
apply (*intro-classes*)
prefer *17*
apply (*rule quadrachotomoy*)
apply (*simp-all add*: *multiplicative-inverse primitive-iff bij-inverse*
positive-inf-mult negative-inf-mult trans-number-ordering)
apply (*rule mult-assoc* [*THEN sym*])
apply (*rule mult-commute*)
apply (*simp add*: *trans-number-divison-def*)
apply (*rule distributivity*, *simp*)
by (*auto simp*: *order-le-less*)

end