RefTime = { type: "CLASS", value: "edu.stanford.nlp.time.SUTime$RefTime" }

SimpleTime = { type: "CLASS", value: "edu.stanford.nlp.time.SUTime$SimpleTime" }

InexactTime = { type: "CLASS", value: "edu.stanford.nlp.time.SUTime$InexactTime" }

InexactDuration = { type: "CLASS", value: "edu.stanford.nlp.time.SUTime$InexactDuration" }

DurationWithFields = { type: "CLASS", value: "edu.stanford.nlp.time.SUTime$DurationWithFields" }

//Duration = { type: "CLASS", value: "edu.stanford.nlp.time.SUTime$Duration" }

IsoDate = { type: "CLASS", value: "edu.stanford.nlp.time.SUTime$IsoDate" }

IsoTime = { type: "CLASS", value: "edu.stanford.nlp.time.SUTime$IsoTime" }

TimeRange = { type: "CLASS", value: "edu.stanford.nlp.time.SUTime$Range" }

TimeWithRange = { type: "CLASS", value: "edu.stanford.nlp.time.SUTime$TimeWithRange" }

RelativeTime = { type: "CLASS", value: "edu.stanford.nlp.time.SUTime$RelativeTime" }

OrdinalTime = { type: "CLASS", value: "edu.stanford.nlp.time.SUTime$OrdinalTime" }

PeriodicTemporalSet = { type: "CLASS", value: "edu.stanford.nlp.time.SUTime$PeriodicTemporalSet" }

ExplicitTemporalSet = { type: "CLASS", value: "edu.stanford.nlp.time.SUTime$ExplicitTemporalSet" }

ANY = -1;

NONE = -1;

NON\_TEMPORAL = "NON\_TEMPORAL";

DURATION\_UNKNOWN = DurationWithFields();

DURATION\_NONE = DurationWithFields();

TIME\_NOW = {

type: REFTIME,

label: "PRESENT\_REF",

value: RefTime("NOW")

}

TIME\_PRESENT = {

type: REFDATE,

label: "PRESENT\_REF",

value: InexactTime( TimeRange(TIME\_NOW, TIME\_NOW) )

}

TIME\_PAST = {

type: REFDATE,

label: "PAST\_REF",

value: InexactTime( TimeRange(TIME\_UNKNOWN, TIME\_NOW) )

}

TIME\_FUTURE = {

type: REFDATE,

label: "FUTURE\_REF",

value: InexactTime( TimeRange(TIME\_NOW, TIME\_UNKNOWN) )

}

// Predefined durations: YEAR, MONTH, DAY, WEEK, HOUR, MINUTE, SECOND

FORTNIGHT = { type: "TIMEUNIT", value: Duration(WEEK, 2) };

HALFHOUR = Duration(MINUTE, 30);

QUARTERHOUR = Duration(MINUTE, 15);

// Basic dates/times

MONDAY = DayOfWeek(1);

TUESDAY = DayOfWeek(2);

WEDNESDAY = DayOfWeek(3);

THURSDAY = DayOfWeek(4);

FRIDAY = DayOfWeek(5);

SATURDAY = DayOfWeek(6);

SUNDAY = DayOfWeek(7);

WEEKEND = {

type: DAYS\_OF\_WEEK,

label: "WE",

value: TimeWithRange(TimeRange(SATURDAY, SUNDAY, Duration(DAY, 2)))

}

JANUARY = MonthOfYear(1);

FEBRUARY = MonthOfYear(2);

MARCH = MonthOfYear(3);

APRIL = MonthOfYear(4);

MAY = MonthOfYear(5);

JUNE = MonthOfYear(6);

JULY = MonthOfYear(7);

AUGUST = MonthOfYear(8);

SEPTEMBER = MonthOfYear(9);

OCTOBER = MonthOfYear(10);

NOVEMBER = MonthOfYear(11);

DECEMBER = MonthOfYear(12);

// Dates are rough with respect to northern hemisphere (actual

// solstice/equinox days depend on the year)

SPRING\_EQUINOX = {

type: DAY\_OF\_YEAR,

value: InexactTime( TimeRange( IsoDate(ANY, 3, 20), IsoDate(ANY, 3, 21) ) )

}

SUMMER\_SOLSTICE = {

type: DAY\_OF\_YEAR,

value: InexactTime( TimeRange( IsoDate(ANY, 6, 20), IsoDate(ANY, 6, 21) ) )

}

FALL\_EQUINOX = {

type: DAY\_OF\_YEAR,

value: InexactTime( TimeRange( IsoDate(ANY, 9, 22), IsoDate(ANY, 9, 23) ) )

}

WINTER\_SOLSTICE = {

type: DAY\_OF\_YEAR,

value: InexactTime( TimeRange( IsoDate(ANY, 12, 21), IsoDate(ANY, 12, 22) ) )

}

// Dates for seasons are rough with respect to northern hemisphere

SPRING = {

type: SEASON\_OF\_YEAR,

label: "SP",

value: InexactTime( SPRING\_EQUINOX, QUARTER, TimeRange( MARCH, JUNE, QUARTER ) ) }

SUMMER = {

type: SEASON\_OF\_YEAR,

label: "SU",

value: InexactTime( SUMMER\_SOLSTICE, QUARTER, TimeRange( JUNE, SEPTEMBER, QUARTER ) )

}

FALL = {

type: SEASON\_OF\_YEAR,

label: "FA",

value: InexactTime( FALL\_EQUINOX, QUARTER, TimeRange( SEPTEMBER, DECEMBER, QUARTER ) )

}

WINTER = {

type: SEASON\_OF\_YEAR,

label: "WI",

value: InexactTime( WINTER\_SOLSTICE, QUARTER, TimeRange( DECEMBER, MARCH, QUARTER ) )

}

// Time of day

NOON = IsoTime( 12, 0, NONE )

MIDNIGHT = IsoTime( 0, 0, NONE )

MORNING = {

type: TIME\_OF\_DAY,

label: "MO",

value: InexactTime( TimeRange ( IsoTime( 6, NONE, NONE), NOON ) )

}

AM = IsoTime(NIL, NIL, NIL, NIL, HALFDAY\_AM)

PM = IsoTime(NIL, NIL, NIL, NIL, HALFDAY\_PM)

AFTERNOON = {

type: TIME\_OF\_DAY,

label: "AF",

value: InexactTime( TimeRange ( NOON, IsoTime( 18, NONE, NONE) ) )

}

EVENING = {

type: TIME\_OF\_DAY,

label: "EV",

value: InexactTime( TimeRange ( IsoTime( 18, NONE, NONE), IsoTime ( 20, NONE, NONE) ) )

}

NIGHT = {

type: TIME\_OF\_DAY,

label: "NI",

value: InexactTime( TimeRange ( IsoTime( 19, NONE, NONE), IsoTime ( 5, NONE, NONE) ) )

}

SUNRISE = {

type: TIME\_OF\_DAY,

label: "MO",

modifier: "EARLY"

}

SUNSET = {

type: TIME\_OF\_DAY,

label: "EV",

modifier: "EARLY"

}

DAWN = {

type: TIME\_OF\_DAY,

label: "MO",

modifier: "EARLY"

}

DUSK = {

type: TIME\_OF\_DAY,

label: "EV",

modifier: "EARLY"

}

LUNCHTIME = InexactTime( TimeRange( IsoTime(12, NONE, NONE), IsoTime(14, NONE, NONE) ))

TEATIME = InexactTime( TimeRange( IsoTime(15, NONE, NONE), IsoTime(17, NONE, NONE) ))

DINNERTIME = InexactTime( TimeRange( IsoTime(18, NONE, NONE), IsoTime(20, NONE, NONE) ))

DAYTIME = {

type: TIME\_OF\_DAY,

label: "DT",

value: InexactTime( TimeRange(DAWN, SUNSET) )

}

MORNING\_TWILIGHT = {

type: TIME\_OF\_DAY,

label: "MO",

value: InexactTime( TimeRange(DAWN, SUNRISE) )

}

EVENING\_TWILIGHT = {

type: TIME\_OF\_DAY,

label: "EV",

value: InexactTime( TimeRange(SUNSET, DUSK) )

}

// For now, just have TWILIGHT be same as EVENING\_TWILIGHT (could possibly be MORNING\_TWILIGHT || EVENING\_TWILIGHT)

TWILIGHT = EVENING\_TWILIGHT

// Relative days

YESTERDAY = RelativeTime( Duration(DAY, -1) );

TOMORROW = RelativeTime( Duration(DAY, +1) );

TODAY = RelativeTime( THIS, DAY );

TONIGHT = RelativeTime( THIS, NIGHT );

HOURLY = PeriodicTemporalSet(NIL, HOUR, "EVERY", "P1X");

NIGHTLY = PeriodicTemporalSet(NIGHT, DAY, "EVERY", "P1X");

DAILY = PeriodicTemporalSet(NIL, DAY, "EVERY", "P1X");

MONTHLY = PeriodicTemporalSet(NIL, MONTH, "EVERY", "P1X");

QUARTERLY = PeriodicTemporalSet(NIL, QUARTER, "EVERY", "P1X");

YEARLY = PeriodicTemporalSet(NIL, YEAR, "EVERY", "P1X");

WEEKLY = PeriodicTemporalSet(NIL, WEEK, "EVERY", "P1X");