Adding some Interpreted Integer Arithmetic to the TPTP

Stephan Schulz, Geoff Sutcliffe

The (new) TPTP syntax has features that were designed to facilitate the addition of interpreted functions and arithmetic. Specifically, (i) interpreted functors and predicates have their own name space, and (ii) numbers have their own name space, and are assumed to be distinct. The relevant extracts from the syntax are:

```
<defined atom>       ::= $true | $false |
                         <term> = <term> | <term> != <term>
<defined term>       ::= <number> | <string>
%----All numbers are implicitly distinct
<integer>            ::= <sign><unsigned integer>
<sign>               ::= + | - | <null>
<unsigned integer>   ::= <0-9>+
```

Now that the syntax has stabilized, following feedback after its first release with TPTP v3.0.0, it seems now possible to make some initial proposals for taking advantage of those features. This proposal is for interpreted functors and predicates on integers in their pure mathematical incarnation, rather than some specific implementation such as 16 bit 2's complement. Five interpreted functors, and one interpreted predicate, are proposed for the TPTP syntax:

- `$plus_int/2` - Addition of two integers.
- `$minus_int/2` - Subtraction of two integers.
- `$times_int/2` - Multiplication of two integers.
- `$divided_by_int/2` - Division of two integers. This is a partial function, being undefined for zero divisors. Non-exact division rounds towards zero.
- `$uminus_int/1` - Unary minus of one integer.
- `$less_int/2` - Comparison of two integers.

The `_int` suffix on each function indicates that it takes integer arguments and returns an integer. This will enables differetiation in the future from analogous functions for, e.g., reals. These will have to be added in the syntax:

```
<defined atom>       ::= $true | $false | <int atom> | ...
<int atom>           ::= $less(<term>,<term>)
<defined term>       ::= <number> | <string> | <int term>
<int term>           ::= $plus(<term>,<term>) | $minus(<term>,<term>) |
                         $times(<term>,<term>) | $divided_by(<term>,<term>) |
                         $uminus(<term>)
```

The extent to which ATP systems are able to work with these may vary, from a simple ability to evaluate ground terms, e.g., `$plus(2,3)` may be evaluated to `5`, through to the ability to instantiate variables in equations involving such functions, e.g.,
`$times(2,$uminus(X)) = $uminus($plus(X,2))`
may instantiate `X` to `2`, to decision procedures for fragments of the syntax.

At this stage all the functions will be written in prefix form. The general issue of flexible specification of infix operators in the TPTP syntax is an orthogonal issue, which must be addressed separately.