P4 Revision Changeset

Proposed version 1.1.0 rc1

Leo Alterman, Barefoot Networks // 08/12/15

This document lays out the concrete changes made in this revision. Sections in the old and new versions of the document are referenced with the text "old §…" and "new §…", respectively.

Goals

This revision of the P4 spec addresses the primary goals Barefoot laid out in its proposal:

- Allow blackbox objects to extend the language
- Allow user code composability and modularity
- Separate architecture from core language specification
- Simplify the core language specification and pull most of the non match+action functionality into a standard library

In addition to the above, a few pieces of syntactic sugar were added to facilitate easier programming and a few pre existing inconsistencies we found in the spec were resolved.

High-level Changes

- "**Whiteboxes**" were introduced, equivalent to "modules" in our original proposal. They are primarily discussed in new §10
- "**Blackboxes**" were introduced, equivalent to "objects" in our original proposal. They are primarily discussed in new §11
- **Types** were introduced, to be used in action/whitebox/blackbox-method signatures. These types are more or less what was already in the language, now stated explicitly.
- A **standard library** was created to include primitives, black boxes, parser exceptions, and other objects that are useful regardless of the target architecture being used. Most of the non-Match+Action objects in the old spec were moved here. The standard library is defined in new §12.

  The standard library is still a work-in-progress and is intended to be evolved over the next few months, especially as features like programmable deparsers are discussed in the language. Its contents are mostly direct translations of objects from the old spec, some of which are due for a reimagining.
- A **Simple Switch Architecture** was created based on the abstract machine in the old P4 spec. All of the figures and passages describing packet flow and high-level P4 operation were moved into here, as well as primitives controlling packet flow. It is defined in new §13.

  The Simple Switch Architecture is still a work-in-progress and is intended to be evolved over the next few months. Its contents are mostly direct translations of objects from the old spec, some of which are due for a reimagining.

Specific Changes

- §1 "Introduction" was rewritten to introduce the concept of target architectures but be otherwise agnostic to the architecture in use
- Old §1.3-1.5 were split off into the new §2 "Structure of the P4 Language"
    - The width specifier for numeric constants was changed from a **'** to a **w** to interact better with the C preprocessor and syntax highlighting
    - Subsections describing typing rules, object references, and expressions were added. These are expanded from fragments of text throughout the old spec, and are the groundwork to allow typed action/blackbox/whitebox signatures and hierarchical references (eg, *a.b.c.d*).
        - min() and max() functions were added to the expression language
        - bit<N> is the primary datatype
        - int is a datatype for situations where the bit width doesn't matter (for example, compile-time parameters)
    - Combined old §15.7 and other fragments about signed/saturating arithmetic and bit width conversions into new §2.4. This area was problematic in the old spec, is still underspecified after this patch, and could benefit from some discussion.
- General grammar changes
    - The grammar used to directly specify the type of reference necessary in each place (eg, extract() expects a "header_name", apply() expects a "table_name"). Most of these have been changed to a generic "object_ref" grammar node that can point to any object, and a semantic rule has been added below the grammar specifying the type required. This was primarily to allow hierarchical object references.
    - Grammar nodes that required at least one element (like header fields and field lists) were changed to also allow zero elements. An argument could be made that allowing "empty" objects makes the act of disabling sections of code easier, and semantic rules could be added where it truly doesn't make sense to have an empty object.
- Headers
    - Header fields are now declared with "type field_name;" syntax (new §3)
        - Most fields are bit<N>
        - Variable width fields (old syntax: "field_name : *") now use the varbit<max_length> type. The length attribute of headers still exists, but the max_length attribute has been moved into the varbit type itself.
    - Added syntactic sugar for declaring temporary/one-use local variables (new §3.2)
    - Added structs (new §3.3) as discussed in proposal
- Moved old §3 "Checksums and Hash-value generators" to standard library (new §12.4).
    - Converted field_list_calculation and calculated_field to equivalent blackboxes
- Parsers
    - Parsers can now invoke other parsers as subroutines, as proposed (new §4.5)

  - Parsers now return 'accept' instead of a control flow function. The target architecture determines where control moves afterwards. This was necessary for parser subroutines and target architecture separation.
  - Parsers can just transition to parser exceptions now, they don't need to use the special 'parse_error' keyword
  - Old §4.3 "Value sets" were moved to the standard library in new §12.7 "Dynamic parser branches". The actual blackbox implementation of value sets is still TBD.
  - Old §4.6.1 "Standard parser exceptions" were moved into the standard library, new §12.2
- Deparser text was modified a bit and a TODO was left indicating that the mechanism should be made programmable, and it needs work.
- Old §6 "Standard Intrinsic Metadata" has been moved into the Simple Switch Architecture, new §13.2
- Old §7 "Counters, Meters, and Registers" were moved into the standard library, §12.3. Their language constructrs were converted into equivalent blackboxes, with their corresponding primitive actions changed into blackbox methods.
- Actions/signatures (new §7) - These changes are primarily about actions, but also apply to blackboxes and whiteboxes
  - Parameter list was changed to include types and directionality (in/out/inout). Directionality is used to allow the passing of non-referenceable things like numeric constants and to more clearly document what the code interface is (what is read/written/etc).
  - Arguments were changed to allow arithmetic expressions in addition to plain references and constants. This allows much greater flexibility, primarily for blackbox methods, but may be too powerful. TODO: this change wasn't strictly necessary, and could be discussed further.
  - The keyword 'primitive_action' was removed. Primitive action declarations just look like function prototypes now.
- Standard action primitives (old §9.1)
  - Most primitives were moved to either the Simple Switch Architecture or standard library, depending on what exactly they did:
    - Core functionality like add_header(), modify_field(), push()/pop(), etc is in the standard library, new §12.1
    - Functionality regarding packet forwarding and replication (clone(), recirculate(), drop(), etc.) is in the Simple Switch Architecture, new §13
    - generate_digest() was turned into a blackbox in the standard library, new §12.6
    - Arithmetic primitives like add_to_field() were removed, since arithmetic expressions now be used in the source of modify_field()
  - The old spec's informal method of specifying primitives and their typing requirements was replaced with a code block for each primitive, specifying its P4 action signature and an accompanying docstring with equivalent information.
- Tables

    - Grammar was changed to allow table attributes to be specified in any order
    - Replaced action_profile attribtue with more general 'modifier' attribute, which allows a table to point to special blackboxes that modify its behavior/define its implementation. Whether this attribute should be called 'implementation' or 'modifier' is TBD
        - Old §10, action profiles and selectors, were merged into one blackbox and moved into the standard library, new §12.5. They now use the table 'modifier' attribute. The text in new §12.5 is copy-and-pasted from the old spec, and could use a rewrite to be more intuitive.
- Control flow
    - Added a 'return' statement which allows a control flow to exit at any point in its body. There is no return value, so this is of limited (but still practical) use.
- Added section on whiteboxes, new §10
    - Includes section on prototypes, which mainly describes how target architectures define programmable regions of code.
- Added section on blackboxes, new §11
    - Also added sections to control flow, parser, and action grammars to allow black box method calls
    - Blackbox methods do not currently have a return type, and cannot be used in expressions. Temporary variables and getter methods are expected to be used instead.