P4 Language Design Working Group 2020-2022

July '20 Meeting

Attendees

- Nate Foster (Cornell)
- Mihai Budiu (VMware Research)
- Hardik Soni (Cornell)
- Hesham ElBakoury (Innovax Technologies)
- Alan Lo (NVIDIA)
- Andy Fingerhut (Intel, Barefoot)
- Chris Dodd (Intel, Barefoot)
- Fabian Ruffy (NYU)
- George Zhou (TUM)
- Gordon Brebner (Xilinx Labs)
- Jed Liu (Intel, Barefoot)
- Stefan Heule (Google)
- Steffen Smolka (Google)
- Vladimir Gurevich (Intel, Barefoot)
- Nikesh Dalal? (YieldStreet)
- Sothy Shan (Nokia)
- Jerimias Blendin (TU Darmstadt)

Status

- We released P4$_{16}$ version 1.2.1! As the semantic versioning suggests, this was a minor revision that clarified some aspects of the specification and added a few features. See the Revision Log for details.
- We are now considering bigger changes to better support:
  - Modularity
  - Abstraction
  - Richer functionality at the edge

Discussion

Issue p4-spec/#2444

- The p4c implementation seems too flexible with respect to infinite-precision integer literals and boolean casts
- The root cause is likely that constant folding does not pay attention to the int type
- Action items: @mbudiu-vmw will draft a fix to the spec

Issue p4-spec/#864

- P4 currently lacks projection operations on tuples, which renders them fairly useless.
- There was a long discussion about whether tuples were redundant (given the existence of structs) or if the programmer should be able to choose the names of the fields.
- Philosophy of adding new types
  - Usually no run-time cost
  - Does add cost for compiler writers and programmers (the "Perl" phenomenon)
  - Also need to consider the control-plane API
- Note that p4c currently forbids using a tuple as a table key
- More broadly...
  - tagged unions
  - arrays?
- Action items:
  - Everyone should look at proposal for tuple projection
  - Vladimir will look at anonymous struct types
  - Nate/Andy will look at tagged unions

Issue p4-spec/#808

- Proposal is to add conditional statements (if-then-else) to parsers
- There is a cost, adding new states.
- And the current p4c implementation has a bug with local variables
- Action Item:
  - Andy will carry this forward.

Issue p4-spec/#656

- Introduces a kind of syntactic sugar for creating overlays or "views" of data types, like headers and structs
- Maybe inspired by Wadler's "Views": https://www.cs.tufts.edu/~nr/cs257/archive/phil-wadler/views.pdf
- There are questions about the restrictions needed to make this sensible if overlays are l-values...

Next Meetings

- July 20th – Brainstorming
- August 1st – "Real" Design meeting

August '20 Meeting

Attendees

- Mario Baldi (Pensando)
- Mihai Budiu (VMware)
- Chris Dodd (Barefoot)
- Andy Fingerhut (Barefoot)
- Nate Foster (Cornell)
- Vladimir Gurevich (Barefoot)
- Jed Liu (Barefoot)
- Fabian Ruffy (NYU)
- Chris Sommers (Keysight)

Announcements

- Shifting to inclusive terminology
- https://github.com/github/renaming
- Nate will take on coming up with a plan for the shift.

Forbid aliasing for out arguments

- Inspired by using Rust, @mbudiu-vmw was wondering about whether we could simply forbid aliasing for out parameters?
- @jafingerhut asked if we could run this on some representative programs to understand the impact
- Example
  Given

```
  void sqrt(out bit<32> res, out bit<32> arg)
```
the invocation
```
   sqrt(a,a)
```
would be forbidden?
@mbudiu-vmw: yes, they would need to create a temporary and do the copying themselves.
@jliu-intel: this seems to be hard for developers of library writers...

- @jnfoster: what's the benefit for doing this?
  @mbudiu: it simplifies the semantics of the language
- Brief discussion of Petr4, a formal semantics developed by Cornell team.
- @jafingerhut: warning on aliasing seems good; programmers may be confused for copy-out and order of arguments, etc.
- @jliu-intel:
  - Has seen real-world code that gets this wrong; bug filed that turned out to be a programming error.
  - Also hard for library writers
- Action item:
  - @mbudiu-vmw will enable alias analysis, with mandatory warnings on out-to-oout and optional on in-to-out...
  - Would be nice to have a -strict flag, like in Perl and other languages.

Issue p4-spec/#880

- The proposal has a detailed writeup of a problem that occurs with exceptional situations in parsers, and several possible fixes
- @jafingerhut: note that this applies not just to extract, but actually to any modification
- @jnfoster: is there a way to write a "careful" parser that always has a defined behavior and never loses bits?
  - @vgurevich: yes, you can use _atomic or _greedy variants, which will result in very slow code.
- @vgurevich: remember that P4 was designed for high-speed hardware, much of which is parallel in nature; may need to relax the sequential semantics of the language
- @jliu-intel:
  - Could add an enum to control the mode (atomic, greedy, fuzzy, etc.)
  - Or, a method to modify the mode of packet_in
  - @vgurevich: that gets tricky; common case is to use a single mode
  - @jliu-intel: another idea is to attach the modes to parser states
- Architectures have broad latitude to define the semantics of reject/accept in Section 12.3 of the spec.
- But we could still add more methods to packet_in...
- @jafingerhut: how does the architecture know what to "muck up"?
- Action Items:
  - @vgurevich to streamline the proposal to reflect what we've learned today...

Issue p4-spec/#883

- Adds a new method to packet_in to initialize a packet buffer from a varbit
- Action Item:
    - Please take a look at Mihai's pull request and make comments

Issue p4-spec/#884

- Proposal is to add overloading based on the types of arguments
- Writeup contains several nice examples where overloading makes the code nicer.
- But concerns about predictability, especially with extensions
- @cdodd: could make ambiguity an error...

Next meetings

- Brainstorming meeting next week
    - p4-constraints
    - Heman's parsers
    - P4++ from VMware?
- September 7th

September '20 Meeting

Attendees

- Jeremias Blendin (Barefoot)
- Gordon Brebner (Xilinx)
- Mihai Budiu (VMware)
- Hesham ElBakoury (Huawei)
- Andy Fingerhut (Barefoot)
- Nate Foster (Cornell)
- Vladimir Gurevich (Barefoot)
- Mary Hogan (Princeton)
- Jed Liu (Barefoot)
- Steffen Smolka (Google)
- Craig Stevens (Dell EMC)

Open Business

- It would be nice to have more folks involved with the open-source compiler, both for tasks like reviewing pull requests and also contributing code.

- Next meeting is October 5th.

Mary Hogan Presented P4All

Discussion of Open Issues

p4-spec/#715

- Does the spec have anything to say whether the order of annotations on an object matters?
- Do we have any annotations whose relative order matters?
- Well, we have free-form annotations so it's hard to say
- A simple solution could be to say that the order of annotations matters
- Action Item:
    - Mihai will take this one.
    - Andy will check for standard annotations.

p4c/#2501

- Proposal adds a mechanism for renaming actions and keys.
- There was broad support for this proposal.
- However, there were some concerns about the concrete syntax.
- Action item:
    - Vladimir will consider concrete syntax and make PRs on the spec.

p4-spec/#887

- This proposes to generalize switch statements so they are more broadly applicable.
- It is also useful for the type-safe union proposal.
- We discussed what kinds of values can be discriminated in a switch statement.
- Would it make sense to add value sets, like parser select statements?
- The proposal also includes a translation to existing match-action tables
- Action item:
    - Andy will create PRs on these...

October '20 Meeting

Attendees

- Mihai Budiu (VMware)
- Andy Fingerhut (Barefoot)
- Antonin Bas (VMware)
- Chris Dodd (Barefoot)

- Hesham ElBakoury (Huawei)
- Nate Foster (Cornell)
- Jed Liu (Barefoot)
- Stefan Heule (Google)
- Vladimir Gurevich (Barefoot)

News and Updates

- It would be fun to get the Lyra [SIGCOMM '20] authors to give us an update on how they are using P4 and other languages like NPL.
- There are now several semantics for P4 being developed (Gauntlet, Petr4, and some others) and interest in cross-validating them and developing machine-readable semantics. This is cool!
- NVIDIA announced a new chip and SDK with P4 support: https://developer.nvidia.com/blog/programming-the-entire-data-center-infrastructure-with-the-nvidia-doca-sdk/

Discussion of Open Issues

p4spec/#887

- There was a lingering question about whether the last case in a switch statement should require a body. Andy resolved this in the spec to be consistent with the current implementation.
- Allowing arbitrary expressions as switch labels makes the grammar ambiguous.
- We resolved to fork the grammar to allow more general expressions as labels.

p4spec/#892

- The proposal seeks to add generic type declarations, e.g., for structs.
- This is also useful for unions...
- Vladimir pointed out that we should have a compelling real-world use case for these extensions. Otherwise it makes the language more complicated.

Union Types

- We had a long discussion of Andy F's initial design and the merits of union types for conserving resources.

p4spec/#877

- Proposal adds names for tuple fields.

Next Meeting

November 2nd.

November '20 Meeting

Attendees

- Mario Baldi (Pensando)
- Antonin Bas (VMware)
- Mihai Budiu (VMware)
- Chris Dodd (Intel)
- Nate Foster (Cornell)
- Andy Fingerhut (Intel)
- Vladimir Gurevich (Intel)
- Stefan Heule (Google)
- Jed Liu (Intel)
- Fabian Ruffy (NYU)
- Konstantin Weitz (Google)

Announcements

- Andy Fingerhut is the TST representative to the LDWG

Process

- Mihai proposes that we spend at most 30 minutes in each meeting trying to resolve issues.
- We decided to do this, but will use the last 30 minutes.
- Stefan noted that it's hard to see what the status is from the list on the Wiki.
- Nate proposed to split the list into two:
    - A long list of items we've started working on
    - A shorter (3-6 items) prioritized list with higher expectations about the goal, status, etc.
- Andy wondered if we could make a standard format for the summary.
- Proposal for comment:
    - Topic:
    - Related issues: <links to other issues or PRs, etc.>
    - Owners:
    - Status:
        - Proposed:
        - Discussed:

        - Implemented:
        - Adopted:
    - Next steps:
    - Discussion:
- Mihai proposed having two people tagged with each review:
    - Owner / Advocate
    - Shepherd / Reviewer

Discussion of Current Issues

p4spec/#892

- This proposal allows generics to be used with structures and other types.
- It's been implemented in the compiler, and is eliminated in the front-end.
- Andy F asked: how would these types work in the control plane?
- Nate F asked how generic structs work with respect to type equality.
    - Currently the spec implies that structs are only equal if they have the same name.
    - The implementation takes care to only generate a monomorphic instantiation once for each type.
- Nate F asked how Gauntlet would handle this?
    - Currently it crashes because Gauntlet has its own type inference and it doesn't handle this feature.
- Vladimir volunteered to be the shepherd for this issue.

p4spec/#887

- We discussed this issue previously.
- There was a small issue with the implementation that has now been resolved.

p4spec/#884

- We discussed this issue previously.
- There was some support for adding this -- it seems useful for programmers.
- Mihai expressed concern that this could be a large change for the compiler.

Open Discussion

- Andy summarized some preliminary work on adding support for modifying table entries in the data plane.
- The Barefoot DPDK backend is a nice contribution. We should have had this a while ago.

Next Meeting

- December 7th

December '20 Meeting

Attendees

- Mihai Budiu (VMware)
- Chris Dodd (Intel)
- Hesham ElBakoury (Nortel Networks)
- Andy Fingerhut (Intel)
- Nate Foster (Cornell)
- Stefan Heule (Google)
- Jed Liu (Intel)
- Alan Lo (NVIDIA)
- Anton Makarov (Gateflow)
- Venkat Pullela (OpenNets)
- Rich Renner (Sunder Networks)

General Type-Safe Unions

- Issue: https://github.com/p4lang/p4-spec/issues/896
- @mbudiu-vmw implemented @jafingerhut's design and, in the process, simplified it
- Discussion:
    - Allowing overwriting the union value with the same tag, within a switch statement
    - The implicit None complicates the design and especially pattern matching
    - Do unions have an exhaustiveness check?
    - We should check how unions work as in/out parameters in calls
- Next steps:
    - Folks should comment on the issues above

Generic Structures

- Issue: https://github.com/p4lang/p4-spec/issues/892
- This feature allows structs, and other data types, to contain generic members.
- It's already been implemented as an experimental feature in the compiler.
- Generic types are eliminated early in the front-end...
- A question arose about the P4Runtime API:
    - If S<bit<8>> appears, the API contains S_0
- More testing is needed but this change sounds good
- We declared it approved by the P4 LDWG

Elimination Form for Tuple Types

- Issue: https://github.com/p4lang/p4-spec/issues/864
- This feature proposes the notation t.f0, t.f1, etc. for the fields in a tuple
- We declared it approved by the P4 LDWG
- @jnfoster will review the p4-spec PR.
- Further discussion on the PR has expressed a preference for array notation.

Conditional Statements in Parsers

- Issue: https://github.com/p4lang/p4-spec/issues/902
- There is a possible generalization with multiple transitions / conditionals in transitions. We deferred this to a future feature.
- We declared it approved by the P4 LDWG
- Some work is needed to fix bugs in the implementation.

Further Generalization to Switch Statements

- Issue: https://github.com/p4lang/p4-spec/issues/890
- This feature generalizes switch statements so they can be used with other types.
- There was broad agreement this is a useful feature.
- We declared it approved by the P4 LDWG
- This feature is already implemented as an experimental feature in the compiler
- Some minor changes are needed to the p4-spec PR; @jafingerhut will fix them

Logistics

- Next meeting January 11th
- Nate will get date for spring P4 events from ONF; we'll plan for v1.3.0 in early spring

January 11, 2021

Attendees

- Mihai Budiu (VMware)
- Chris Dodd (Intel)
- Andy Fingerhut (Intel)
- Nate Foster (Cornell)
- Stefan Heule (Google)
- Jed Liu (Intel)
- Alan Lo (NVIDIA)
- Anton Makarov (Gateflow)

- Vladimir Gurevich (Barefoot)
- Mario Baldi (Pensando)

Notes:

Merged changes since last meeting:

Allow if statements in parser states

Generalize switch statements to allow bit<W>, enum, and similar expressions

Issue: Tuples: no elimination form

- Not closed yet because of a proposal to use array index syntax, e.g. tuple_name[0], tuple_name[1] instead of compiler-synthesized field names tuple_name.f0, tuple_name.f1, etc.
    - Vladimir: So far in P4 array-like things (only header stacks today?), all elements have the same type. That would not be true if we use this syntax for tuples.
    - Control plane APIs - does P4Runtime API let one specify values of tuples? Yes, as one of the cases allowed in P4Data message type.
- Mihai: Requires a different compiler IR to implement array index syntax, but that is not showstopper.
- Array indexes must be compile-time known values, so compiler can statically determine type of tuple element.

Issue #904 (p4c issue #2444): Casting integers to bools

- There are multiple choices we could make for the example code in the text of the Github issue.
- Nate: Concern: If there are two ways to cast between two values, a language spec should give some thought to guaranteeing that all ways will give the same result.
- Mihai: Note that the current spec doesn't say much about a statement like "const int a = 2w1;" and what that should do.
- Seemed to be consensus that we should allow cast from int to bool, but only when value of int is 0 (->false) or 1 (->true).
- Also to allow assignments from values of type bit<W> and int<W> to int.

Generic Structures (p4-spec issue #892):

- The implementation technique used by p4c today for generic types is to eliminate names like S<bit<32>> and replace them with auto-generated named S_0.
    - This is an issue if we ever want to refer to type names like S<bit<32>> in a control plane API.
- Another approach is to make P4 use structural typing for structs (two types considered the same/compatible if structs have same field types, perhaps in the same order, and perhaps with same field names), rather than today's nominal typing that is implemented in open source p4c (same name of struct type only, not merely same "structure").
    - That could be done everywhere in the language, or more conservatively, only for these types auto-generated from generic types. This more conservative approach is what is part of Mihai's PR for p4c.

What does same type mean? (p4-spec issue #875):

- Definitely relates to issues raised for Generic Structures.

General-purpose safe unions:

- Mihai: Discovered a problem with my previous proposal. Code inside of switch statements can call actions that modify the unions. This seems to require interprocedural analysis in p4c to detect and prevent.
- One way to avoid this: make union values immutable.
- Mihai proposed a syntax for copying the currently-valid union member in each switch branch, that avoids the need to do such interprocedural analysis. Looking for comments on it.
    - Inside of a switch statement, the variable that has type "union" can only be used as an L-value, not as an R-value. The local names created can be used as L-value or R-value, but exist only in the lexical scope of that branch, and so need not have any interprocedural analysis.

Description of abstract methods (p4-spec issue #771):

- Chris: TNA does still want to use the capability of abstract methods that read or write variables in lexical scope, e.g. for at least some RegisterAction extern definitions.
- Andy: My main objection before was whether the pure function restriction was useful for anyone, in any way. As long as we don't stop at pure functions and work towards generalizing to functions that can access variables in lexical scope, perhaps with @synchronous annotation to restrict when the abstract method can be called, that seems reasonable.
- Mihai: Look at current proposal with restriction for pure functions.

February 1, 2021

Review of previously discussed issues that have been closed

- Casts between numeric types and bools
  https://github.com/p4lang/p4c/issues/2444

Brief discussion of previously discussed issues

- Generic structures
  https://github.com/p4lang/p4-spec/issues/892
- Type-safe unions
  https://github.com/p4lang/p4-spec/issues/896
- Abstract methods
  https://github.com/p4lang/p4-spec/issues/561

Main discussion items

- Tuples

    - Support for one-element tuples
      https://github.com/p4lang/p4-spec/issues/908
    - Elimination forms for tuples with array-like indexing
      https://github.com/p4lang/p4-spec/issues/864
- Initialization syntax and semantics
    - Header Stacks
      https://github.com/p4lang/p4-spec/issues/198
    - Support for initializing invalid headers
    - Does it matter if you do/don't initialize struct-like objects with no fields

New discussion items, time permitting

- Order of Annotations
  https://github.com/p4lang/p4-spec/issues/715
- Statically-Unrolled Loops
  https://github.com/p4lang/p4-spec/issues/907
- Support for parsing varbits
  https://github.com/p4lang/p4-spec/pull/883

February 1, 2021

Attendees

- Mario Baldi (Pensando)
- Mihai Budiu (VMware)
- Chris Dodd (Barefoot)
- Vladimir Gurevich (Barefoot)
- Nate Foster (Cornell)
- Stefan Heule (Google)
- Jed Liu (Barefoot)
- Alan Lo (Nvidia)
- Ricardo Parizotto (UFRGS)

Review of Previously Discussed Items

- Casts between numeric types and bools
  https://github.com/p4lang/p4c/issues/2444

  The implementation is done. We agreed to add one sentence to the specification explaining that true is 1 and false is 0. Nate will do this.

Main discussion items

- Tuples
    - Elimination forms for tuples with array-like indexing
      https://github.com/p4lang/p4-spec/issues/864

      We were previously considering two different syntaxes for tuple access: t.f1, t.f2, vs t[0], t[1], etc.

      Vladimir asked whether overloading [.] was a good idea: unlike header stacks accessing tuples can return a different type and also requires a compile-time known value (which is important for static checking)

      The PR for the compiler had a large change because the def-use analysis changed.

    - We also discussed the semantics of structs with no fields in the case where they are uninitialized. We decided that no spec changes were needed, but clarifying what equivalence means when it is empty could be helpful.

      For the implementation, we decided to change the def-use analysis to suppress warnings when the size in bits of a data type is 0.

    - Support for one-element tuples
      https://github.com/p4lang/p4-spec/issues/908

      We briefly discussed this issue as being approved by the LDWG, with Vladimir as a supporter.

New discussion items, time permitting

- Statically-Unrolled Loops
  https://github.com/p4lang/p4-spec/issues/907

  Andy expressed surprise that we'd tie ourselves more to the C preprocessor

  Vladimir pointed out that you can use m4

  Chris pointed out that you can do loops with BoostPP

  Nate pointed out that we discussed this a long time ago:
  https://github.com/p4lang/p4-spec/issues/84

  Seems like we can split two issues: (i) a general pre-processor / macro-like system (ii) a bounded for loop solution

  Stefan pointed out that a complicated pre-processor can lead to problems for tools.

- Order of Annotations
  https://github.com/p4lang/p4-spec/issues/715

Progress Update on Older Items

- Type-safe unions
  https://github.com/p4lang/p4-spec/issues/896

  Jed proposed a scheme for naming the field members of type-safe unions `f as x:`

  Mihai proposed to make this mandatory

March 1, 2021

Attendees:

* Chris Dodd (Barefoot)
* Nate Foster (Cornell/Barefoot)
* Mihai Budiu (VMware)
* Chris Sommers (Keysight)
* Ben Pfaff (VMware)
* Fabian Ruffy (NYU)
* Rich Renner (Sunder Networks)
* Vladimir Gurevich (Barefoot)
* Anton Makarov (Gateflow)
* Andy Fingerhut (Barefoot)
* Antonin Bas (VMware)

Announcements

* Fabian is now a committer
* Next meeting: scheduled for April 5, 2021
* P4 workshop: May 18-20, 2021

Discussion

* Issue 914: https://github.com/p4lang/p4-spec/issues/914

The description in the spec for action arguments is no longer accurate.
It does not handle const entries or default actions which can supply action arguments in the program itself (and not from the control-plane, as stated).
Consensus: to be reviewed and merged. Approved.

* PR #913: https://github.com/p4lang/p4-spec/pull/913

Boolean serialization
Data types with no bits.
We should add bit<0> in the spec. We should allow a 0 to initialize a bit<0> value. To be done as a future PR against the spec. Larger values can be assigned to width 0 bitstrings and will cause an overflow warning. Mihai to submit a PR for bit<0>.
PR should be merged.

* PR #888: https://github.com/p4lang/p4-spec/pull/888
Tuples with a single element
Andy asked why the grammar can't be simpler
Mihai to validate the grammar in the PR

* PR #902: https://github.com/p4lang/p4-spec/pull/902
If statements in parsers
This can be merged as soon as the compiler fixes are implemented. Nate asked for an additional test in the compiler implementation. Once that is done we can merge and close this issue. AI: Mihai should write the additional test.

* Writing tests for Tofino Native Architecture, brought up by Vladimir
Ideally Intel should contribute the tna.p4 to p4include.

* Issue #864: https://github.com/p4lang/p4-spec/pull/864
Tuple elimination form
We settled on t[0] syntax.
We have discussed this, but since last time we fixed the compiler to no longer give warnings for empty data structures. AI: fix typo, merge the two PRs in compiler and spec and close the issue.

* Issue #915: https://github.com/p4lang/p4-spec/issues/915
What is the type of header stack indexes?
Fix greater than to greater than or equal.
How are errors handled by P4? Out of bounds indexes?
How about out of bounds indexes in a stack in the parser?
Vladimir: when architectures subset the language, how do we find out what is available?
Will file two issues: error handling requirements, and how to discover what is available in an arch?
Nate and Andy will write a PR.

* issue 896: https://github.com/p4lang/p4-spec/issues/896
safe unions
Andy and Ben will followup with Mihai
Please comment on the spec PR

* issues 883 and 264: https://github.com/p4lang/p4-spec/pull/883,
https://github.com/p4lang/p4-spec/issues/264
Getting data out of varbits
Should we have remaining_length() for packet?

April 5, 2021

Attendees

- Chris Dodd (Barefoot)
- Nate Foster (Cornell/Barefoot)
- Mihai Budiu (VMware)
- Andy Fingerhut (Barefoot)
- Vladimir Gurevich (Barefoot)
- Stefan Heule (Google)
- Ben Pfaff (VMware)
- Rich Renner (Sunder Networks)
- Anton Makarov (Gateflow)

Review previously merged changes

- Empty data, serialization of Booleans: https://github.com/p4lang/p4-spec/pull/913
- Action data description: https://github.com/p4lang/p4-spec/issues/914
- Tuples with a single element: https://github.com/p4lang/p4-spec/pull/888, https://github.com/p4lang/p4-spec/issues/908
- Generic structures: https://github.com/p4lang/p4-spec/issues/892
- If statements in parsers: https://github.com/p4lang/p4-spec/issues/902
- Tuple field access: https://github.com/p4lang/p4-spec/issues/864, https://github.com/p4lang/p4-spec/pull/877

New Issues

- Updates to P4-16 grammar: https://github.com/p4lang/p4-spec/pull/928
    - Mihai will review these changes (e.g., void)
    - Chris to explore why forward declarations are needed.
- Clarifications for parameters: https://github.com/p4lang/p4-spec/pull/861
    - Andy F to add cross reference to compile-time known values.
- State and repeated invocations: https://github.com/p4lang/p4-spec/issues/926
    - Ryan Doenges to create an example illustrating the semantics in this case
- Change serEnum grammar: https://github.com/p4lang/p4-spec/pull/866

    - Generalizes the proposal for serializable enums; has already been merged into p4c.
- Reserved identifiers: https://github.com/p4lang/p4-spec/issues/929
    - We discussed several designs:
        - Reserving some identifiers (e.g., starting with "__")
        - Adding an annotation (e.g., "@preserved")
        - Adding a more explicit way to indicate the architecture
- Initialize invalid headers by assigning the value ‘false’: https://github.com/p4lang/p4-spec/issues/341
    - We discussed several syntax including false, {}, and invalid.
    - Adding invalid would require grabbing a new keyword.

Fixed made to previous issues

- Packet length for PSA: https://github.com/p4lang/p4-spec/pull/927 or https://github.com/p4lang/p4-spec/pull/925
    - We are leaning toward not adding a length field -- it seems hard to do in a portable and precise way, moreover with cut-through processing.
- Add match_kind optional to PSA: https://github.com/p4lang/p4-spec/pull/920
    - We are deferring this question for now. PSA can add optional.
- Type of stack indexes: https://github.com/p4lang/p4-spec/pull/923, https://github.com/p4lang/p4-spec/issues/915
    - Andy F to check whether serializable enums work, otherwise we approved this change.
- Zero-width bitstrings: https://github.com/p4lang/p4-spec/pull/931
    - There were no objections, so we approved this change.
- Initializing header stacks: https://github.com/p4lang/p4-spec/issues/198
    - To finalize a design we need:
        - Concrete syntax
        - Semantics, including edge cases (e.g., length mis-match, .next and .last)
        - We should do this... it seems useful and completes the language design.
    - Mihai to start an implementation

May 3, 2021

Attendance

- Mario Baldi (Pensando)
- Mihai Budiu (VMware)
- Chris Dodd (Barefoot)
- Andy Fingerhut (Barefoot)
- Nate Foster (Cornell)
- Paul Gazzillo (University of Central Florida)
- Alan Lo (NVIDIA)

- Anton Makarov (3A Alliance)
- Denis Matousek (Netcope)
- Ben Pfaff (VMware)
- Fabian Ruffy (NYU)

Upcoming Events

- P4 Workshop: run by ONF: https://opennetworking.org/2021-p4-workshop-3-0/
- P4 1.2.2 release will be done soon (before the workshop)

Spec changes merged since last time

- Change serEnum grammar: https://github.com/p4lang/p4-spec/pull/866
    - Generalizes the proposal for serializable enums
    - has already been merged into p4c.
- PSA packet length
    - Adopted explanation of why length is not available;
    - https://github.com/p4lang/p4-spec/pull/927
    - closed alternative proposal 925

Changes that are ready to merge for 1.2.2

- Types of stack indexes: https://github.com/p4lang/p4-spec/pull/923
    - fixed p4c bug for serenum indexes: https://github.com/p4lang/p4c/pull/2738
    - To be merged as is for 1.2.2
- Zero width bitstrings: https://github.com/p4lang/p4-spec/pull/931
    - In spec added varbit<0> as well.
    - Compiler support merged in https://github.com/p4lang/p4c/pull/2724
    - To be merged for 1.2.2
    - Replace "positive or zero" with non-negative
- Updates to P4-16 grammar: https://github.com/p4lang/p4-spec/pull/928
    - Checked the status of changes
    - "void" was introduced as a legal type argument by Chris Dodd 3 years ago
    - We decided to remove the forward extern declaration from the spec and keep as experimental in the compiler. AI: Andy Fingerhut to change the spec

Old issues revisited

- Process to change master to main for p4c compiler: https://github.com/p4lang/p4c/pull/2708
    - Travis did not check this
    - AI: Nate to merge as is using superuser powers and we'll fix it it it breaks
- Type-based method disambiguation: https://github.com/p4lang/p4-spec/issues/884

    - Proposal to reject ambiguous calls is more reasonable
    - Not clear we have resources to explore this soon

New issues

- action_run when default action is not specified: https://github.com/p4lang/p4-spec/issues/933
    - Change "may" to "must" in spec
- Generic typedefs: https://github.com/p4lang/p4-spec/issues/934
    - Should prototype and we'll evaluate later
- Architecture-independent libraries
    - stdlib: factor out common code between psa.p4 and pna.p4: https://github.com/p4lang/p4-stdlib
- KV Tables API: https://github.com/p4lang/p4c/pull/2739
    - Based on unions; are unions really necessary?

Older issues

- Safe Unions: https://github.com/p4lang/p4-spec/issues/896
    - AI: Mihai to write a description of the current design
    - Based on pattern matching
    - Fully working in compiler
- Abstract methods: https://github.com/p4lang/p4-spec/pull/771
    - No support for variable capture yet
    - We will attempt to merge in 1.2.2

June 14, 2021

Attendees

- Nate Foster (Cornell)
- Mihai Budiu (VMware)
- Andy Fingerhut (Intel)
- Youri Lavoie (Intel)
- Vladimir Gurevich (Intel)
- Fabian Ruffy (NYU)
- Kwan Kim (Dell)

Logistics

- Next meeting: August 9th

- We'll organize a series of summer working meetings (no changes to the spec approved) on other dates.
    - Nate: July 12th

Agenda

Fix the p4-spec page to show the latest release.

- sizeof() for scalars
  https://github.com/p4lang/p4-spec/issues/937

  - Mihai: it's somewhat strange that this operation can be applied to values and types
  - Nate: I'd like to see the design rationalized with the minSizeInBits() method of headers
  - Vladimir: is it a problem that these are not objects? Mihai: no.
  - We agreed to try the method approach, and to explore generalizing to other types.

  Action Item:
  - Vladimir will craft a spec proposal
  - Mihai will work on the implementation in the compiler

- Serializable enums for value sets
  https://github.com/p4lang/p4-spec/issues/868

  The current specification is overly restrictive in what types of elements may appear in a parser value set. For example, serializable enums (along with bools, and some other types) are not supported.

  Andy Fingerhut will move the current sentence out of the section on `select` and into a section on value sets. He will also rationalize the set of types supported and the keys of tables. He will also check if the implementation of ranges and other operations are working or have bugs.

- Syntax to assign names to actions
  https://github.com/p4lang/p4-spec/issues/900

  We previously discussed the importance of being able to choose control-plane names (suggesting against a @name annotation).

  We discussed the desire to have a common syntax for many forms of renaming: actions, keys, arguments, etc.

  Vladimir desires the syntax to come *before* the thing being renamed.

Vladimir will ponder the syntax and develop a design.

- Add methods to packet_in object for parser exception model
  https://github.com/p4lang/p4-spec/issues/880

  The current language spec says that an extract that fails must leave the packet buffer in an unmodified state, and transition to the reject state.

  See for example:
  https://github.com/p4lang/p4c/blob/main/testdata/p4_16_samples/parser_error-bmv2.p4
  https://github.com/p4lang/p4c/blob/main/testdata/p4_16_samples/parser_error-bmv2.stf

  We ran out of time, so this discussion will be continued at a future meeting.

Issues deferred until next time...

- Reserved Identifiers
  https://github.com/p4lang/p4-spec/issues/929
- Match kind set
  https://github.com/p4lang/p4-spec/issues/795
- Type-Safe Unions
  https://github.com/p4lang/p4-spec/issues/896
- Namespaces and imports
  https://github.com/p4lang/p4-spec/issues/718

September 13, 2021

Attendees
- Jeremias Blendin (Intel)
- Nate Foster (Cornell / Intel)
- Ben Pfaff (VMware)
- Bola Malek (Rivian)
- Andy Fingerhut (Intel)
- Mario Baldi (Pensando)
- Steffen Smolka (Google)
- Jonathan DiLorenzo (Google)
- Kishan Shanmugam (Google)
- Anthony Dalleggio (JP Morgan)
- Fabian Ruffy (NYU)
- Ali Kheradmand (Google)
- Denis Matousek (Brno University of Technology)

Agenda

- Announcements
    - Discussion of plans for version 1.2.3
    - P4 TST Elections
- Recirculation on V1Model
  https://github.com/p4lang/p4c/issues/2875
    - The initial design for P4_16 couldn't accommodate the semantics of recirculation in V1Model.
    - We hoped that PSA's design would be adopted, but as of 2021, V1Model is still in widespread use.
    - For example, the clone3 primitive takes as an argument a set of lvalues to be preserved on non-linear packet paths.
    - There was an older proposal that used @field_list annotations on the type declarations that supports the P4_14 behavior.
    - We noted that standard_metata would need to be copied to a programmer-defined type.

- sizeof() for scalars
  https://github.com/p4lang/p4-spec/issues/937
    - We have previously discussed the value of having a "sizeof"-like operator in the language.
    - P4_16 has a method minSizeInBits() that can be applied to header instances; we'd like to extend it to other serializable types.
    - Using sizeof() would be non-backward compatible.
    - So there are two proposals:
        - Extend the current .minSizeInBits() method to other serializable expressions *and* their types
        - Add a new symbol like <#>
    - We observed that in C-like languages, it's hard to get the size of a member of a struct if you don't have an expression of that type
    - **Action item:** Andy Fingerhut will write up proposals using both concrete syntax
- Clarifications and bugfixes raised by Petr4 Folks
    - https://github.com/p4lang/p4-spec/issues/954
        - The spec is silent on what kinds of types of expressions are allowed in select expressions.
        - Action Item: Andy Fingerhut will add a sentence clarifying the allowed types (bit<...>, int<...>, bool, lists, etc.)
    - https://github.com/p4lang/p4-spec/issues/955
        - The spec is not fully precise about what kinds of expressions are allowed as indexes for slices.
        - Action Item: Nate will add a sentence clarifying the allowed types (bit<...>, int<...>, int) and write tests.
    - https://github.com/p4lang/p4-spec/issues/956
        - The description of the concatenation operation is confusing.

        - Action Item: The Petr4 team will add sentences to the spec to clarify.
    - https://github.com/p4lang/p4-spec/issues/957
        - The specification should clarify that shifts do not lead to implicit casts.
        - Action Item: The Petr4 team will add a sentence to fix the spec, and the broken example.
    - https://github.com/p4lang/p4-spec/issues/958
        - The spec is not fully definitive on the status of implicit casts for serializable enums.
        - Action Item: The Petr4 team will study what p4c does and formulate a proposed revision to the spec.
    - https://github.com/p4lang/p4-spec/issues/959
        - The spec is confusing on whether bit types can be used on the right hand sides of shifts.
        - Action Item: The Petr4 team will study what p4c does and formulate a proposed revision to the spec.
    - https://github.com/p4lang/p4-spec/issues/960
        - The spec does not say whether lists can be used in equality operations.
        - The Petr4 team will add a sentence to the specification
    - https://github.com/p4lang/p4-spec/issues/961
        - This question concerns casts between list and struct expressions.
        - Action Item: The Petr4 team will add an illustrative example to the spec.
    - https://github.com/p4lang/p4-spec/issues/962
        - The spec doesn't list value sets in the control-plane names section.
        - Action Item: the Petr4 team will add it to the spec.
- Parser exceptions
  https://github.com/p4lang/p4-spec/issues/880
    - Mihai has a simpler proposal based on new extract methods and error-checking methods.
    - Action Item: Vladimir to weigh in on the simpler proposal and well seek to find a design that works for his target and is simple and minimally disruptive.
- Generalized varbits
    - https://github.com/p4lang/p4-spec/issues/901
        - This proposal lets you treat a varbit as a packet_in object
        - We discussed whether these packet_in objects can be invoked in controls; if so, then the semantics would need to change
        - Does it make sense to add a new extern instead?
    - https://github.com/p4lang/p4-spec/issues/264
        - This proposal adds new operations on varbits, such as dynamic length, varbits, etc.
        - Perhaps we need some applications to drive this proposal forward.
- Open Business
    - Please help review PRs on the front-end, or better, get involved with hacking!

October 4, 2021

Attendees

- Mihai Budiu (VMware)
- Nate Foster (Cornell/Intel)
- Ali Kheradmand (Google)
- Andy Fingerhut (Intel/Barefoot)
- Ben Pfaff (VMware)
- Steffen Smolka (Google)
- Kapil Agrawal (UC Irvine)
- Jonathan DiLorenzo (Google)
- Fabian Ruffy (NYU)
- Chris Dodd (Intel/Barefoot)

Agenda

- Nominations for Steering Committee are closed
- How should we evolve the spec? Can we do 2 releases/year?
- Recap on how changes to the spec are made

minSizeInBits / minSizeInBytes https://github.com/p4lang/p4-spec/issues/937

- Is this the right syntax?
- Is the definition right for stacks? How about unions? The name is confusing in these cases. Perhaps tuples should be there as well.
- Perhaps we should not define this method for unions and stacks.
- Perhaps we should not define it for varbit.
- Should it be hidden for newtype types?
- We could name this method 'sizeof' as well.
- This gives an error for enums. This should be mentioned.

What types should be allowed in select expressions?
https://github.com/p4lang/p4-spec/pull/968

- The PR as written is missing some types: enum, bool, and list types.
- What about headers/structs/varbits?
- This may break our backwards compatibility.
- Structs should work.
- Should we review the section on set types in the spec?
- We should try the minimal set of types we know are needed.
- Let's also check what backends support.

Fixing recirculation: https://github.com/p4lang/p4c/pull/2902

- Steffen Smolka approves of this PR
- These operations should be renamed
- What should the name be?
- clone_with_fieldlist is a good pattern

Discussion on status of PSA/DPDK backends

varbits parsing: https://github.com/p4lang/p4-spec/issues/901

- who needs this?
- under some circumstances it could be completely eliminated by the compiler
- could be useful for TLV parsing
- could be done symmetrically to write into varbits using packet_out
- open problem: what else would people want to use P4 for?

Open issues

- Andy Fingerhut: ability to apply a table from an action. Why not?
- switching on action_run could be used to implement this. This would work if the action does not use conditionals.
- How about support for conditionals in actions?
- Could controls have public tables that can be invoked from somewhere else?
- How about invoking the same table twice?
- How about an interpreter build in a control that can invoke dynamically tables based on its invocation arguments?
- Google has use cases where tables are invoked multiple times. They are not real tables.

November 1, 2021

Attendees

- Mihai Budiu (VMware)
- Antonin Bas (VMware)
- Nate Foster (Cornell & Intel)
- Andy Fingerhut (Intel)
- Steffen Smolka (Google)
- Jonathan DiLorenzo (Google)
- Mario Baldi (Pensando)
- Vladimir Gurevich (Intel)
- Chris Dodd (Intel)
- Fabian Ruffy (NYU)
- Jeremias Blendin (Intel)

Community

- We discussed ways to encourage more people to contribute to the compiler.
- Ideas:
    - Improve documentation in the compiler passes
    - Point to documentation more prominently on GitHub
    - Create easy on-ramps for newcomers
    - Gauntlet is an example of an academic project that is maybe slightly imperfect in terms of coverage but has a big impact; Petr4 could do the same, with differential testing.
    - Consider online hackathons / live coding of simple features?
    - Tweak Andy's VM?

minSizeIn{Bits/Bytes}: https://github.com/p4lang/p4-spec/issues/937

- Andy and Vladimir discussed and agreed on 0 as the "min size" for variable-length data like varbits.
- It is legal to apply this operation to "invalid" objects, like invalid headers, or even elements of stacks that are out of bounds
- We also want a "max size" operation on headers.

Type	**minSizeInBits**	**maxSizeInBits**
bit<N>	N	N
int<N>	N	N
bool	1	1
enum bit/int<N> { ... }	N	N
enum { ... }	Undefined	Undefined
string	Undefined	Undefined
void	Undefined	Undefined
match_kind	Undefined	Undefined
error	Undefined	Undefined
int	Undefined	Undefined
tuple of types	sum of minSIB(types)	sum of maxSIB(types)

varbit<N>	0	N
struct with fields	sum of minSIB(fields)	sum of maxSIB(fields)
header with fields	sum of minSIB(fields)	sum of maxSIB(fields)
h_t[N]	N * minSIB(h_t)	N * maxSIB(h_t)
header union h1 h2	max(minSIB(h1), minSIB(h2))	max(maxSIB(h1), maxSIB(h2)
controls/parsers/ packages	Undefined	Undefined

minSIB(T) is:

- defined on all serializable types (or vice versa)
- in the absence of varbits, minSIB and maxSIB are identical
- in the presence of varbits, maxSIB is the worst-case size of the serialized representation of the data and minSIB is the "best" case.

```
header h0 { bit<8> x; bit<0> y; }
header h1 { bit<8> x; bit<1> y; }
header_union u1 { h0 f0; h1 f1; }

header h { bit<8> x; varbit<1> y; }

minSIB(u) = max(8, 9) = 9
maxSIB(u) = max(8, 9) = 9

minSIB(h) = 8;
maxSIB(h) = 9;
```

We agreed to take this table, even though some felt the minSizeInBits semantics is somewhat odd.

December 2021

Attendees

- Molly Pan (Princeton University)
- Nate Foster (Cornell/Intel)
- Mihai Budiu (VMware)
- Alan Lo (NVIDIA)

- Dan Talayco (Individual)
- Fabian Ruffy (NYU)
- Qinshi Wang (Princeton University)
- Vladimir Gurevich (Intel)
- Andy Fingerhut (Intel)
- Paul Gazzillo (University Central Florida)
- Sanjeeva Yerrapureddy (World Wide Technologies)
- Ali Kheradmand (Google)
- Chris Dodd (Intel)

What is the right license for the P4 spec?

Re-announcing the office hours P4 bi-weekly meeting

Clarifying the spec

minSizeInBytes and other variants https://github.com/p4lang/p4-spec/issues/937

- recap the previous discussion
- ready to merge

type of bit slicing https://github.com/p4lang/p4-spec/pull/970

- types of indexes for slicing can be broader than originally stated
- should specify that values are non-negative
- this should be merged

types used in concatenation https://github.com/p4lang/p4-spec/pull/980

- concatenation was not listed as an operation of fixed-width values
- this was merged, fixed issue 956

implicit casts https://github.com/p4lang/p4-spec/pull/981

- clarifications for shifts and concatenation
- make the example even more explicit
- should be merged after fixes
- fix the wording of the << 13 example

implicit casts for serializable enums https://github.com/p4lang/p4-spec/pull/982

- Should the slice of an enum be an l-value?
- Should we allow slicing on enum values that keeps them l-values?
- These should be handled in a separate PR
- Should be reworded and then it will be accepted

shifts by fixed-width values https://github.com/p4lang/p4-spec/pull/983

- shifts by 0 are allowed
- Moved shift section around and added one more paragraph
- fix the negative example
- should be accepted

equality comparisons https://github.com/p4lang/p4-spec/pull/984

- clarify all types that support == and !=
- Removed section on structure initializers
- Will discuss again, too long

control plane names https://github.com/p4lang/p4-spec/pull/986

- merged during the meeting

We will have a winter release of P4, but we'll attempt to merge all these clarification PRs before

January 2022

January 3, 2022.
We are using a new zoom meeting link starting today.
The address is
https://cornell.zoom.us/j/97583070442?pwd=UHVEMmUzMTVXb1lDWkhPSHM0TTRuUT09

Attendees

- Molly Pan (Princeton University)
- Nate Foster (Cornell/Intel)
- Mihai Budiu (VMware)
- Qinshi Wang (Princeton University)

- Andrew Pinski (Marvell)
- Lin Ma (Marvell)
- Andy Fingerhut (Intel/Barefoot)
- Vladimir Gurevich (Intel/Barefoot)

We are continuing the discussion of small spec fixes
There will be a P4 workshop in the spring.

Consistent naming: https://github.com/p4lang/p4-spec/issues/1004 and https://github.com/p4lang/p4-spec/issues/762

- Justified by the casing for pop_front
- We should not remove any names from the language
- Add new names for these methods/fields as long as they are local
- Andy Fingerhut will own the naming part
- Vladimir will own fields/methods

Have a method to return the type of a stack https://github.com/p4lang/p4-spec/issues/978

- We do not have expressions that return types
- Could do probably easily if only used within a typedef
- This could be a much more general method to get the type of any expression, like C++ typeof
- Let's postpone this feature for now

Constructor parameters are not compile-time known values https://github.com/p4lang/p4c/issues/2995 and https://github.com/p4lang/p4c/issues/1001

- Templated controls would solve this
- Modular type checking is difficult
- Related to https://github.com/p4lang/p4c/issues/932
- Could have the typing stages expressed in the spec
- Petra has a simpler formalization of the type checking, perhaps we should adopt that

Clarify uninitialized values with examples https://github.com/p4lang/p4-spec/issues/988

- Uninitialized values can be read multiple times and provide different results

- But "uninitializedness" does not propagate
- The definition of copy-in and copy-out implies assignment
- Copy propagation seems to be incorrect currently since it can make programs less deterministic

We will have another meeting on January 24 to continue the discussion

Jan 24, 2022 Extra Meeting

- Discuss some open issues and pull requests on p4-spec
- https://github.com/p4lang/p4-spec/issues/977
    - This seems like a feature used by some P4 compilers, and an oversight it was left out of the spec.
    - Qinshi: Does this have implications for the type checker?
    - Nate: Later in the same section modified is the phrase “The implementation of such objects is not expressed in P4”, but parsers and controls have behavior that IS expressed in P4, so we may need to think through this change more carefully.
        - I would like to write some more tests to see how robust the current p4c implementation is in this regard right now.
        - I would be suspicious of type-checking logic and overload resolution logic in regards to this proposed change.
    - Nate volunteered to write a few test programs to see if he can break the type checker in this area.
- https://github.com/p4lang/p4-spec/pull/989
    - Qinshi: Are the sentences mentioning direct type invocation too restrictive?
    - Molly: Given the text in the section on direct type invocation, that sentence is trying to be consistent with it.
    - Suggested change to one of the sentences near the end of the addition.
    - AI Andy: Will add a comment with a suggested alternative to the two paragraphs in question.
        - Something like: “All of the above applies regardless of whether an instantiation is explicit, or done via direct type invocation (with ref to section on DTI).”
- https://github.com/p4lang/p4-spec/pull/985
    - Plan to merge this one. Seems like a small clarification.
- https://github.com/p4lang/p4-spec/pull/984
    - Note that a significant change here is replacing all occurrences of structure initializers with structure-valued expressions. It seems that this was the intent when adding structure-valued expressions that structure initializers should become obsolete.
    - Some discussion on the sentence “A list expression can be compared with a list expression, …”. Nate added in-line comments to the PR during the meeting.

    - Nate: If we have a header and a tuple in a P4 program with `==` or `!=` to compare them, what does p4test do today with this? Does it add casts? He wrote a program to try running through p4test.
        - Some impressive hacking by Nate and runs of p4test by Radostin back and forth. :-)
    - Nate: "I'd be curious how the type system is doing this." (referring to his example program, now added to the issue)
    - Mihai: Today, a list expression is implicitly cast to a struct expression or a header, when you compare it via == to a value of those types.
    - Nate: Based on Mihai's answer, I think we should instead explain how implicit casts are added, and when, onto list expressions, and explain it that way in the spec. Nate added a comment to the PR during the meeting.
    - Molly: I will check my test programs in the issue I created for this to confirm.
    - Some discussion on whether it would be safe/well-defined/predictable to take and expression like this "{a = 1, b = 2}", and arbitrarily pick some struct type with fields named a and b, inspired by this comment: https://github.com/p4lang/p4-spec/issues/960#issuecomment-984005608
    - Nate: It seems that current p4c effectively implements "header-valued expressions", similar to the currently specified structure-valued expressions. The spec already mentions this.
    - Nate added a comment to suggest that this PR should be refactored into at least two, one that removes mentions of structure initializers. The other suggested change involved adding an explanation of the rewrites that the p4c compiler is doing in some of these programs, since they aren't exactly implicit casts, but something else.
- Mihai asked others to look at the two issues he created on p4-spec most recently.
    - Some discussion on if locally scoped type declarations cause problems, and if so, why. I did not attempt to capture the discussion here.

February 7 Meeting

Attendance
- Nate Foster (Cornell/Intel)
- Andy Fingerhut (Intel/Barefoot)
- Vladimir Gurevich (Intel/Barefoot)
- Qinshi Wang (Princeton)
- Radostin Stoyanov (Oxford)
- Mihai Budiu (VMware)
- Ali Kheradmand (Google)
- Molly Pan (Princeton)
- Chris Dodd (Intel/Barefoot)
- Fabian Ruffy (NYU/Intel)

Announcements from Andy Fingerhut

- P4 has developer days presentations every two days. Next meeting is about the compiler architecture.
- The Portable Network architecture meetings will be public soon.

Next meeting on March 7, 2022.

Exit execution restrictions: https://github.com/p4lang/p4-spec/pull/860

What happens if exit is called in an action that is invoked from a table that appears in an expression? Three variants: expression produces an undefined value, if used in a RHS then the LHS is unchanged, or forbid it altogether.
AI: we should merge this after a couple of more approvals.

Capitalization of names: https://github.com/p4lang/p4-spec/issues/1004

We need a linter to enforce these rules. Otherwise they are too difficult.
For now this issue will be closed.

This in abstract methods: https://github.com/p4lang/p4-spec/issues/973

Externs are an escape hatch, so perhaps this should be allowed. This should be a decision of the architecture. So we'll close this issue without changes to the spec. Spec change: add more columns to the tables in Appendix F of the spec, about abstract methods.

More restrictions on abstract methods https://github.com/p4lang/p4-spec/issues/976

We still don't have a description of synchronous or asynchronous annotations.
This problem in this issue does not seem to be important. Target architectures and compilers will choose whether these constructs are supported. Same solution as for 973. We are closing 802 as well, related and now obsolete. This issue will be closed.

Remove instantiations from block for issue 975: https://github.com/p4lang/p4-spec/pull/1006

This should be accepted: instantiations only allowed outside an apply block. Merge this. We sould merge the corresponding one in the compiler.

Optional parameters in parser and control types: https://github.com/p4lang/p4-spec/issues/977

These optional parameters are already allowed in parsers and controls.
How about optional parameters in control and parser types? Nate will write a draft to solve this problem.

Local typedefs: https://github.com/p4lang/p4-spec/pull/1011

and typeof keyword: https://github.com/p4lang/p4-spec/pull/1012
Simple and useful features to consider for addition. Please review.

March 7, 2022 Meeting

Attendees

- Andy Fingerhut (Intel)
- Alan Lo (NVIDIA)
- Fabian Ruffy (NYU)
- Molly Pan (Princeton)
- Qinshi Wang (Princeton)
- Radostin Stoyanov (Oxford)
- Venkat Puella (Keysight)
- Vladimir Gurevich (Intel)
- Dan Talayco (Intel)

Issues

- Syntax for invalid header
  https://github.com/p4lang/p4-spec/issues/1021

  Section 8.23 already provides the "..." syntax for initializing with default values. So we can already initialize a single header with
  H h = ...
  or a header nested within a struct as
  S s = { h = ..., g = 0 }
  We should clarify though if this is allowed.

- Removing structure initializers (since we now have structure-value expressions)
  https://github.com/p4lang/p4-spec/issues/1021

  Several attendees reviewed this item and agreed it's a good change that simplifies the language specification. After these approvals were added, we merged the PR during the meeting.
- Improve and move operations on sets
  https://github.com/p4lang/p4-spec/pull/1008

  This seems like a good change: it clarifies possible discrepancies between the p4c implementation and the spec, and it also introduces a new concept of numeric types that could streamline some aspects of the spec.

  However, we decided to table it for now, until some questions about how types of operations (e.g., "&&&") are resolved and the interaction with implicit casts.

- Specify that slices of serializable enums are valid l-values:
  https://github.com/p4lang/p4-spec/issues/992

  Previously we observed a problem with slices of serializable enums. In particular, because they had to go through an implicit cast, such slices were not l-values.

  We agreed that slicing should be an operation on serializable enums. This side-steps the problem with implicit casts. We also agreed to add a small clarification on when slices are l-values. @MollyDream will author a PR.
- Semantics of header unions
  https://github.com/p4lang/p4-spec/issues/995

  We reviewed the history of header unions, and their support (or lack thereof) on existing proprietary P4 compilers.

  The issue here concerns the behavior of statements like, u.h1.setInvalid(), which also sets all other headers in the union to be invalid.

  Dan Talayco observed that validity is really a property of containers, and a union is a container. So manipulating the valid bits for headers within a union may be somewhat strange thing to do.

  After this brief discussion, we tabled this item for a future meeting.

Meeting of April 2022

Attendees

Nate Foster (Cornell/Barefoot)
Mihai Budiu (VMware)
Venkat Pullela (Keysight)
Qinshi Wang (Princeton)
Radostin Stoyanov (Oxford)
Andy Fingerhut (Intel)
Dan Talayco (Intel)
Molly Pan (Princeton)
Fabian Ruffy (NYU/Intel)
Alan Lo (NVIDIA)
Jonathan DiLorenzo (Google)
Mario Baldi (Pensando)

Function side effects as action arguments (https://github.com/p4lang/p4-spec/issues/840, https://github.com/p4lang/p4-spec/pull/852)

Action arguments (in table actions list) can be complex expressions.
This was superseded by https://github.com/p4lang/p4-spec/pull/1043
We will adopt the simplified PR. This one was closed. The simplified one was merged.

Allow switch statements in actions (https://github.com/p4lang/p4-spec/pull/945)

The PR was merged, allowing switch statements as long as they don't cause calling other tables.

Allow serializable enums in ValueSets: https://github.com/p4lang/p4-spec/pull/873

Should we allow bools too?
How about P4Runtime?
Should be resolved by a separate working group and ideally approved separately, as general as possible.

Clarify types allowed in select expressions: https://github.com/p4lang/p4-spec/pull/968

why not struct types?
Should be resolved by a separate working group and ideally approved separately, as general as possible.

Optional parameters: https://github.com/p4lang/p4-spec/pull/987

How does unification with optional arguments work?
Optional parameters behave like (potentially) uninitalized values of the specified type.
Spec does not specify unification at all. We should probably take this PR and if desired write additional spec for how type unification works. The PR has been merged.

Difference between control-level and apply-level declarations: https://github.com/p4lang/p4-spec/pull/989

We should adopt this, but the wording should be improved.

Constructor parameters are compile-time known: https://github.com/p4lang/p4-spec/pull/1002

This is related to the question about templates vs. static type checking. For example, what is the type of a[v:0]? There may be multiple ways to solve this problem: multiple phases of type checking in the compiler, template-based type checking (based on instance only), or be vague.

?: as a compile-time constant: https://github.com/p4lang/p4-spec/pull/1039

This was merged.

typeof operator, and local typedef: https://github.com/p4lang/p4-spec/pull/1012 and https://github.com/p4lang/p4-spec/pull/1011

Members should think about these features and comment on them. We could make typeof more powerful.

May 2022

Attendees

- Mihai Budiu (VMware)
- Nate Foster (Cornell)
- Andrew Pinski (Marvell)
- Fabian Ruffy (NYU)
- Jai Kumar (Broadcom)
- Jeremias Blendin (Intel)
- Dan Talayco (Intel)
- Andy Fingerhut (Intel)
- Alan Lo (NVIDIA)

Allow direct invocation of generic parsers/controls

- Spec PR: https://github.com/p4lang/p4-spec/pull/1069
- Compiler PR: https://github.com/p4lang/p4c/pull/3260
- This PR adds support for direct invoking generic parsers/controls. This seems like a useful feature (Vladimir has some practical examples). We agreed to take it. However, Mihai will add an example in the spec.

Allow don't cares for named arguments:

- Spec PR: https://github.com/p4lang/p4-spec/pull/1074
- Compiler PR: https://github.com/p4lang/p4c/pull/3274
- This PR allows don't care ("_") to be used with named arguments. We believe this has no knock-on effects elsewhere in the language.
- We agreed to merge this PR as it seems like a bugfix.

Do not allow typedef with unspecialized generic types

- Spec PR: https://github.com/p4lang/p4-spec/pull/1076
- Compiler PR: https://github.com/p4lang/p4c/pull/3174
- This PR fixes a hole in the language specification that allowed using typedefs with the name of a generic type but without its argument.
- We agreed to merge this PR as its a bugfix.

Language design commentary

Vladimir observed that it would be nice to have a "commentary document" (distinct from the language spec) that discusses various alternatives and design choices we considered. Maybe someday...

Is 'isValid' a valid header field name

- Spec PRs:
    - https://github.com/p4lang/p4-spec/issues/1060
    - https://github.com/p4lang/p4-spec/pull/1079
- Since methods and field names can never be confused in a well-typed program, we believe this is a useful feature for programmers, so we agreed to merge it.

Explain type inference for constructors

- https://github.com/p4lang/p4-spec/issues/1067
- https://github.com/p4lang/p4-spec/pull/1072
- We agreed to take this clarification (after adding a comma to fix a typo).

Specifying struct types for list expressions

- https://github.com/p4lang/p4-spec/issues/1061
- https://github.com/p4lang/p4-spec/pull/1075
- This PR allows specifying structure types for list expressions

Overloads of parsers

- https://github.com/p4lang/p4-spec/issues/1055
- https://github.com/p4lang/p4-spec/pull/1078
- This PR would forbid overloading of parsers, controls, and packages.
- Nate proposed that we could apply our existing conventions for overloading to parsers/controls. When creating an instante, we would do it for the constructor; in a direct invocation, we would do it for the implied apply method.
- We had a meta-conversation about evolving the spec, compatibility with p4c, etc. And we took a vote, which was 2-1 in favor of accepting the change, but then we later agreed to delay discussion given that this seemed inconclusive.

Behavior of next index for extracting to header unions

- https://github.com/p4lang/p4-spec/issues/1048
- https://github.com/p4lang/p4-spec/pull/1064
- This PR is not yet ready... so we will wait for next time.

Operations available on objects typed by type variables

- https://github.com/p4lang/p4-spec/issues/1049
- https://github.com/p4lang/p4-spec/pull/1077
- This PR adds a new section on operations on type variables, clarifying that there are basically no operations on type variables, other than assignment. We agreed to take this change.

June 2022

Attendees

- - Mihai Budiu (VMware) (on vacation this meeting)
- + Nate Foster (Cornell / Intel)
- + Andrew Pinski (Marvell)
- + Fabian Ruffy (NYU)
- + Dan Talayco (Intel)
- + Andy Fingerhut (Intel)
- + Chris Dodd (Intel)
- + Sandy Frost (Los Alamos National Laboratory)
- + Reshma Sudarshan (Intel)
- + Vladimir Gurevich (Intel)
- + Alan Lo (Nvidia)

Release of version 1.2.3 of language specification

- Nate Foster has created a PR for version 1.2.3 release candidate. Please take a look and add review comments if you have any: https://github.com/p4lang/p4-spec/pull/1110
  - Another question to think about when reviewing the changes since 1.2.2: If you believe the version number should be 2.0.0 or 1.3.0 instead of 1.2.3, add your reasoning in comments.
- Should p4c version number be increased to match latest spec version after spec is released?
  - Andy: Note that default initializers using `...` syntax is in 1.2.2 of the spec, but not implemented yet in latest p4c, so perhaps p4c’s version number should actually be 1.2.1.<something> until that feature is implemented?
  - TODO: Will discuss p4c version numbering and release cadence in 2022-Jun-07 P4.org open source developer days meeting.

Priorities for future LDWG meetings

- Some future-looking proposals that have been discussed in the last several months:

    - Alan Lo presented idea for new P4 syntax and semantics for writing state machines.
    - There has been some progress towards a P4 namespace/module system over last several months. Hope to bring a concrete proposal to LDWG meeting some time soon-ish.
    - Gordon Brebner mentioned work related to generalizing P4 from packets to events.
        - Vlad: Isn't this already effectively present in the language, e.g. if a P4 architecture specified that a P4 control was invoked on particular events that are not packets?
    - Defining a syntax/semantics for defining P4 architectures precisely, i.e. in something like code rather than natural language.
    - Nate: I have a post-doc student working on clean-slate Petr4 implementation of P4. She has been writing a formal model for P4 semantics.
- Some smaller (?) items
    - How to deal with overloading of method calls, and perhaps other things?
    - Vlad's proposal on "less precise" extract methods that promise less if packet data runs out before a complete header is extracted.
    - An idea from some time ago from Steffen Smolka. A P4 program defines two things: a data plane behavior for processing packets, and a control plane API, although this second thing is not described as extensively in the language spec as the first.
        - Inspired by Java interfaces, perhaps, enabling more control over what the control plane API is for a P4 program, e.g.
        - Hide some tables/externs from control plane
        - Something more than @name annotation for giving different control plane API names than the default implied by the source code. Vlad had some thoughts there in a public issue or two.
    - This dovetails with question of how we specify control plane API for P4 extern objects. Currently there is nothing in the P4 language today that lets you specify a control plane API for a P4 extern.
    - There was quite a bit of discussion among Vlad, Dan, Nate on control plane API generation, not captured here.
    - Should we have a way in P4 to specify the control plane API for P4 extern objects? Vlad: Yes, that would be nice.
    - Should we have a way in P4 to specify the control plane API for fixed function components? Vlad: Yes, that would be nice.
    - Should we try to devise standard control plane APIs, even for common things like counters/meters? Vlad: I think that might be difficult, even for common externs like those.
        - Example: Vlad described two similar but different APIs for P4 tables with ternary keys:
            - a P4Runtime-API-like API, with relative numeric priority values given for each entry added, and P4 keys are keys of the control

plane API, and there are add, delete, and modify operations available in control plane API. Note that this API can be used not only for hardware TCAM, but also for many kinds of algorithmic TCAM.

- A closer-to-hardware API that only works with hardware TCAM, where instead of relative numeric priority values, control plane API uses hardware indexes in the TCAM in a small range (the size of the number of hardware entries), validity of each entry is exposed, there are only modify operations, no add/delete operations, and value/mask are part of the “data” of the entry, and only the hardware index is the key.
- Andy: Note that even though multiple control plane APIs exist, note that a vendor can choose to implement both of these, perhaps with at most one of the APIs used for a single P4 ternary table at a time. There is some precedence already in the P4Runtime API spec, where tables with implementation ActionSelector already have two different control plane API options in that one spec, where for a single table, controller software should use at most one style at a time on the same P4 table.

August 2022 Meeting (August 7, 2022, 1PM PT)

Attendees

- Nate Foster (Cornell/Intel)
- Parisa Ataei (postdoc at Cornell)
- Andy Fingerhut (Intel)
- Andrew Pinski (Cavium)
- Sandy Frost (LANL)
- Mihai Budiu (VMware)
- Roop Mukherjee
- Radostin Stoyanov (Oxford)
- Vladimir Gurevich (Intel)

Logistics

- Due to the US Holiday (Labor Day), we'll move the September LDWG to the *second Monday of the month.*
- We have updated the spec README.md file to bring it up-to-date with the latest processes.

Technical Discussion

- Operations on match_kind values
  https://github.com/p4lang/p4-spec/issues/997
  https://github.com/p4lang/p4-spec/pull/1124

  Spec was updated, change was accepted.

- Arbitrary precision vs infinite precision
  https://github.com/p4lang/p4-spec/issues/1084
  https://github.com/p4lang/p4-spec/pull/1125

To add this to change log then merge it. @jafingerhut to make the change to the changelog.

- Type of result of *size* methods
  https://github.com/p4lang/p4-spec/issues/1096
  https://github.com/p4lang/p4-spec/pull/1128

  Spec was updated, change was accepted.

- Semantics of inverted ranges
  Ranges on serializable enums
  https://github.com/p4lang/p4-spec/issues/1104
  https://github.com/p4lang/p4-spec/issues/1103
  https://github.com/p4lang/p4c/pull/3482

  Alternative semantics for inverted ranges would be the complement of the range.
  Change was accepted and merged.
  The related compiler change has been merged as well. It converts an error into a warning - it accepts strictly more P4 programs than before. If we desire a "complement" semantics, we should add a new "complement" operator for sets.
  This change was accepted and merged.

- Static assert
  https://github.com/p4lang/p4-spec/issues/1123
  Discussion about breaking backwards compatibility. Even using a new header file may break compatibility. There is an associated compiler implementation. @mbudiu-vmw has submitted a PR against the spec

- Default action mutability
  https://github.com/p4lang/p4-spec/issues/1113
  https://github.com/p4lang/p4-spec/pull/1126
  https://github.com/p4lang/p4c/pull/3481

We should change the spec and not the compiler. @jnfoster will submit a PR against the spec. The compiler change has been closed. This was later merged.

- Empty keys for tables
  https://github.com/p4lang/p4-spec/issues/1120

We will change the spec to state that an empty key is the same as no key. Andy Fingerhut will submit a PR against the spec. This was later merged.

- Overloading of parsers and controls
  https://github.com/p4lang/p4-spec/issues/1080 and
  https://github.com/p4lang/p4-spec/pull/1078 and
  https://github.com/p4lang/p4-spec/issues/1055

  Vladimir to describe whether this is acceptable from a practical point of view.

- Direct type invocation with constructor arguments
  https://github.com/p4lang/p4-spec/issues/1068
  https://github.com/p4lang/p4-spec/pull/1130

  @mbudiu-vmw should submit a PR against the spec that constructor arguments are not supported for direct invocation. This was subsequently merged.

- Padding types
  https://github.com/p4lang/p4-spec/issues/1121

  Could be also treated as fixed 0 values.

- Addition assignment (+=) operator
  https://github.com/p4lang/p4c/issues/3258
  @rst0git should write a PR against the spec.

September 2022 Meeting (September 12, 2022, 1PM PT)

Attendees

Mihai Budiu (VMware)
Nate Foster (Cornell/Intel)
Andy Fingerhut (Intel)
Andrew Pinski (Cavium/Marvell)
Mostafa Elbediwy (Polytechnique Montreal)
Parisa Ataei (Cornell)
Dan Talayco (Intel)
Ryan Goodfellow (Oxide)
Bili Dong (Google)
Vladimir Gurevich (Intel/Barefoot)
Alan Lo (NVIDIA)
Radostin Stoyanov (Oxford)
Roop Mukherjee (NVIDIA)

Miscellaneous Issues

Links are broken in the posted version of the spec. Stale working draft still exists. ONF is working on these. Next design meeting is scheduled on October 3, Monday.

New language features:

- Static assert
  https://github.com/p4lang/p4-spec/issues/1123
  https://github.com/p4lang/p4-spec/pull/1129

This should be merged later today. It was.

- Support for generic vector/list types and literals
  https://github.com/p4lang/p4-spec/issues/1140
  https://github.com/p4lang/p4c/pull/3520

Probably we should use a syntax similar to struct initializers.

Would be nice to revisit the tuple/list typing issue; Nate to study this issue.

- Compound assignments
  https://github.com/p4lang/p4-spec/pull/1144
  https://github.com/p4lang/p4c/issues/3258
  https://github.com/p4lang/p4-spec/pull/1144

To add logical operations.

These should be single tokens.

We should not have != and ==.

The two paragraphs contradict each other.

We should prototype this in the compiler before accepting it.

Clarifications

- Key names for keys with masks
  https://github.com/p4lang/p4-spec/pull/1141
  https://github.com/p4lang/p4-spec/issues/1138

We are not ready to take this change.

We should verify what various implementations do.

What is the authoritative name? The p4info file?

Action item: @vgurevich will comment on this.

Perhaps we should make these just a suggestion and not mandatory.

An alternative naming scheme just assigns small integers to key fields for a table.

@mbudiu-vmw will edit this section and make it clear that this is just a suggestion.

- Describe core.p4 match kinds semantics
  https://github.com/p4lang/p4-spec/issues/1090
  https://github.com/p4lang/p4-spec/pull/1145

Let's work more on this offline on github.

- Inout vs out parameters
  https://github.com/p4lang/p4-spec/issues/1142
  https://github.com/p4lang/p4-spec/pull/1146

To be read and ideally merged. This was later merged.

- Validity after lookahead
  https://github.com/p4lang/p4-spec/issues/1014 and
  https://github.com/p4lang/p4c/issues/3041
  https://github.com/p4lang/p4-spec/pull/1148

@mbudiu-vmw will handle even more general cases

- Slicing int values
  https://github.com/p4lang/p4-spec/issues/1015
  https://github.com/p4lang/p4-spec/pull/1149

To fix typos. This was later merged.

October 2022 meeting, October 3, 1-2:30PM Pacific Time

Attendees:

Andy Fingerhut (Intel/Barefoot)
Alan Lo (NVIDIA)
Andrew Pinski (Cavium)
Mihai Budiu (VMware)
Mostafa Elbediwy (Politechnique Montreal)
Nate Foster (Cornell/Intel)
Radostyn Stoyanov (Oxford)
Hari Thantry (Google)

Miscellaneous

Next meeting is scheduled on Monday November 7.

- Switching development to C++ 17
  https://github.com/p4lang/p4c/pull/3547
  No objections.

Clarifications

- Overloading of parsers and controls
  https://github.com/p4lang/p4-spec/issues/1080 and
  https://github.com/p4lang/p4-spec/pull/1078 and
  https://github.com/p4lang/p4-spec/issues/1055

We should have a general rule to accept such changes - a "philosophy" of the language. But we can take the PR while we figure out what it is. This change was merged.

- Replace "list expression" with "tuple expression"
  This is intended to allow the new vector/list types to be named "list"
  https://github.com/p4lang/p4-spec/pull/1156

People like this change. Let's give it more time to gather comments, but we should merge it if there are no objections.

- Describe core.p4 match kinds semantics
  https://github.com/p4lang/p4-spec/issues/1090
  https://github.com/p4lang/p4-spec/pull/1145

This needs more work. We'll work on it offline and either merge it or discuss it at the next meeting.

- Clarify the meaning of "const entries"
  https://github.com/p4lang/p4-spec/pull/1155

  We should accept this change once it has a changelog. This was merged.

- Simplify description of annotations
  https://github.com/p4lang/p4-spec/pull/1152

  This should be merged after we check the spelling. Change was merged.

- Clarify types allowed in select expressions
  https://github.com/p4lang/p4-spec/pull/968

This seems a useful change. We should change the changelog but ideally merge it. This was later merged.

- Key names for keys with masks
  https://github.com/p4lang/p4-spec/pull/1141
  https://github.com/p4lang/p4-spec/issues/1138

  This should be merged.

- Validity after lookahead
  https://github.com/p4lang/p4-spec/issues/1014 and
  https://github.com/p4lang/p4c/issues/3041
  https://github.com/p4lang/p4-spec/pull/1148

  This is merged.

New language features

- Compound assignments
  https://github.com/p4lang/p4-spec/pull/1144
  https://github.com/p4lang/p4c/issues/3258
  https://github.com/p4lang/p4-spec/pull/1144

  This is making progress, Radostin is working on an implementation. He will also update the spec PR based on suggestions.

- Support for generic vector/list types and literals
  https://github.com/p4lang/p4-spec/issues/1140
  https://github.com/p4lang/p4c/pull/3520

We should change the name from Vector to 'list' (lowercase) after we take the list->tuple expression PR.

Should we allow a syntax that does *not* specify the type? (Just a tuple expression as an initializer?)

Mihai to check the failing cases submitted by Andy, then to write a PR against the spec that we will discuss next time.

We discussed about various possible syntaxes for the literals. If we use {} then we can rephrase the table entries as having a list<tuple<>> type.

- Initial table entries
  https://github.com/p4lang/p4-spec/issues/1159

Specifying entries for non-core match kinds is already a problem. See the google doc for the complete proposal. We could have a new abstract data type for priorities.

November 2022 meeting, October 7, 1-2:30PM Pacific Time

Attendees:

Vladimir Gurevich (Intel+Barefoot)
Nate Foster (Cornell/Intel+Barefoot)
Andy Fingerhut (Intel+XFG)
Andrew Pinski (Cavium+Marvell)
Radostin Stoyanov (Oxford)
Mihai Budiu (VMware)
Roop Mukherjee (NVIDIA)
Mostafa Elbediwy (Polytechnique Montreal)

Miscellaneous:

We should grant @apinski-cavium permissions to review and modify the spec and the compiler.

Next meeting is scheduled for Monday, December 5, 2022.

Agenda

Clarifications:

- Duplicate table properties
  https://github.com/p4lang/p4-spec/issues/1162
  https://github.com/p4lang/p4-spec/pull/1163

  Andrew should rephrase and we will take the change when it's ready. Was merged.

- Equality for tuples
  https://github.com/p4lang/p4-spec/issues/1094
  https://github.com/p4lang/p4-spec/pull/1178

  This change was accepted and merged.

- Key fields have to have scalar types
  https://github.com/p4lang/p4-spec/issues/1176

  We do not have a design for struct or complex key fields. Should we write one? Should we disallow that? Leave this for now. Vladimir will try to find a real use case.

- Describe core.p4 match kinds semantics
  https://github.com/p4lang/p4-spec/issues/1090
  https://github.com/p4lang/p4-spec/pull/1145

Has been discussed previously; review comments have been addressed. To be reviewed offline and ideally merged before the next meeting.

New features

- List types (was "vector" types)
  https://github.com/p4lang/p4-spec/pull/1156
  https://github.com/p4lang/p4-spec/pull/1168
  https://github.com/p4lang/p4c/pull/3520

  Review comments were addressed. A changelog is needed for these PRs. The first one should be merged. Vladimir would really like a form which has implicit casts. To check for conflicts introduced by the new keyword.

- New direction specifier for action parameters
  https://github.com/p4lang/p4-spec/issues/1177

  To be expanded.

- Literal for invalid headers
  https://github.com/p4lang/p4c/pull/3667

  This was favored, @mbudiu-vmw should write a pull request against the spec.

- Compound assignment
  https://github.com/p4lang/p4c/pull/3669
  https://github.com/p4lang/p4-spec/pull/1144
  https://github.com/p4lang/p4c/issues/3258
  https://github.com/p4lang/p4-spec/pull/1144

We now have an implementation ready. The implementation does not reflect the spec with respect to side-effects. The spec can be accepted before the compiler implementation. We should analyze the spec before the next meeting.

December 2022 meeting, December 5, 1-2:30PM Pacific Time

Attendees:

- Mihai Budiu (VMware)
- Mohsen Rahmati (Politechnique of Montreal)
- Andrew Pinski (Cavium)
- Nate Foster (Cornell/Barefoot)
- Fabian Ruffy (NYU/Barefoot)
- Chris Dodd (Barefoot)
- Thomas Calvert (AMD)
- Andy Fingerhut (Barefoot)
- Hari Thantry (Google)
- Mario Baldi (AMD)

Miscellaneous:

- The google docs document is becoming large and slow. Should we archive the top portion in a separate read-only document?
    - Nate will do it.
- Next meeting falls on Jan 2, perhaps we should do it on Jan 9
- P4 workshop is being planned in person. Should we have in person working group meetings? Are there problems traveling?

Agenda

Clarifications:

- Describe core.p4 match kinds semantics
  https://github.com/p4lang/p4-spec/issues/1090
  https://github.com/p4lang/p4-spec/pull/1145

Ideally should be ready to merge. Andy will review this with Mihai and we'll merge it today.

- Key names for keys with masks
  https://github.com/p4lang/p4-spec/pull/1141
  https://github.com/p4lang/p4-spec/issues/1138

  Has been discussed in the past, but not yet merged. Hopefully we will merge this today. This was later merged.

- type for non-base types
  https://github.com/p4lang/p4-spec/issues/1044
  https://github.com/p4lang/p4-spec/pull/1179

  people should weigh on this topic and we'll merge it when it's ready. This was later merged.

- Types of expressions in select lists
  https://github.com/p4lang/p4-spec/issues/954
  https://github.com/p4lang/p4-spec/pull/968

  Has been discussed previously but not merged. Needs to be fixed. Andy will review and hopefully we'll merge today. This was merged.

- Constructor parameters are compile-time know values
  https://github.com/p4lang/p4-spec/issues/1001
  https://github.com/p4lang/p4-spec/pull/1002/

Has been discussed previously. Nate owns this issue and will lead the future evolution of this proposal.

- Default values for action arguments
  https://github.com/p4lang/p4c/issues/3657

  Please weigh on this topic as well.

- Evaluation order of implicit casts
  https://github.com/p4lang/p4c/issues/3587

  Subtle interaction between two evaluations orders. What does Petra do?

New language features

- Literal syntax to represent the value of "an invalid header"
  https://github.com/p4lang/p4-spec/pull/1184
  https://github.com/p4lang/p4-spec/issues/1021
  https://github.com/p4lang/p4c/pull/3667 (merged)

  Some interesting alternatives were proposed; this will need to cook a bit more.

- Add list type
  https://github.com/p4lang/p4-spec/issues/1194

https://github.com/p4lang/p4-spec/pull/1168
https://github.com/p4lang/p4c/pull/3520 (merged)

See comments: update grammar. We should merge this before the next meeting. This was later merged.

- Define "initial entries" for P4 tables, in addition to currently defined "const entries"
https://github.com/p4lang/p4-spec/issues/1159
https://github.com/p4lang/p4-spec/pull/1180
https://github.com/p4lang/p4c/pull/3748

This is the first time we see this PR. The compiler supports the syntax, but does not yet implement the priority assignment.