Copland Core Language Definition (An extension of MITRE/KU/APL language)

Adam Petz and Perry Alexander

University of Kansas

1 Terms

$$\begin{aligned}
P &\leftarrow \mathit{place} \\
M &\leftarrow \mathit{asp\_id} \\
A &\leftarrow \mathsf{USM}\ M\ \bar{a} \mid \mathsf{KIM}\ M\ P\ \bar{a} \mid \mathsf{SIG} \mid \mathsf{HSH} \mid \mathsf{CPY} \mid \mathsf{NONCE} \mid \cdots \\
t &\leftarrow A \mid @_P\, t \mid (t \rightarrow t) \mid (t \overset{\pi}{\prec} t) \mid (t \overset{\pi}{\sim} t) \\
E &\leftarrow \xi \mid \mathsf{U}_P(E) \mid \mathsf{K}_P^P(E) \mid [\![E]\!]_P \mid \#_P\, E \mid \mathsf{N}_P(E) \mid (E \mathbin{;;} E) \mid (E \parallel E) \mid \cdots
\end{aligned}$$

where $\pi = (\pi_1, \pi_2)$ is a pair of splitting functions and $\bar{a}$ is a list of arguments.

Fig. 1. Term Grammar

2 Concrete Evidence

$$\begin{aligned}
\mathsf{ARG} &\leftarrow \mathit{string} \\
\mathsf{BS} &\leftarrow \mathit{bits} \\
e &\leftarrow \mathsf{mt} \mid \mathsf{U}_P\ M\ [\mathsf{ARG}]\ \mathsf{BS}\ (e) \mid \mathsf{K}_P^P\ M\ [\mathsf{ARG}]\ \mathsf{BS}\ (e) \mid \mathsf{G}_P\ e\ \mathsf{BS} \mid \mathsf{H}_P\ \mathsf{BS} \mid \mathsf{N}_P\ \mathsf{BS}\ (e) \mid \mathsf{SS}\ e\ e \mid \mathsf{PP}\ e\ e \cdots
\end{aligned}$$

Fig. 2. Conrete evidence Grammar

3 Messages

$$\begin{aligned}
M_{ID} &\leftarrow bits \\
REQ &\leftarrow M_{ID}\ \mathsf{P}\ \mathsf{P}\ \mathsf{t}\ \mathsf{e} \\
RES &\leftarrow M_{ID}\ \mathsf{P}\ \mathsf{P}\ \mathsf{e} \\
m &\leftarrow Request\ REQ \mid Response\ RES
\end{aligned}$$

Fig. 3. Messages Grammar

4 Data Exchange Format (JSON Schema)

Every JSON object representing an Alegbraic Data Type(ADT) has two members:

1. “name”-maps to the constructor name string (e.g. "KIM", "K", "Request"). Note: Constructor names should be unique to allow unambiguous parsing.
2. “data”-maps to a JSON array that holds the arguments for that particular constructor(members of that array will differ from constructor to constructor).

4.1 General ADT Schema

```
{
  "name": < string >,
  "data": < array >
}
```

4.2 Request Message Schema

Corresponds to Figure 3

```
{
  "name": "Request",
  "data": [
    < string >,
    < number >,
    < number >,
    < term >,
    < evidence >
  ]
}
```

4.3 Response Message Schema

Corresponds to Figure 3

```
{
  "name": "Response",
  "data": [
    < string >,
    < number >,
    < number >,
    < evidence >
  ]
}
```

4.4 Protocol Term Constructor Schemas

Corresponds to Figure 1

```
{
  "name": "USM",
  "data": [
    < number >,
    [< string >]
    ]
}
```

```
{
  "name": "KIM",
  "data": [
    < number >,
    < number >,
    [< string >]
  ]
}
```

```
{
  "name": "SIG"
}
```

```
{
  "name": "HSH"
}
```

```
{
  "name": "NONCE"
}
```

```
{
  "name": "AT",
  "data": [
    < number >,
    {
      "name" : <T_constructor_name>,
      "data" : [...]
    }
  ]
}
```

```
{
  "name": "LN",
  "data": [
    {
      "name" : <T_constructor_name>,
      "data" : [...]
    },

    {
      "name" : <T_constructor_name>,
      "data" : [...]
    }
  ]
}
```

```
{
  "name": "BRS",
  "data": [
    [<"ALL" | "NONE">, <"ALL" | "NONE">],
    {
      "name" : <T_constructor_name>,
      "data" : [...]
    },

    {
      "name" : <T_constructor_name>,
      "data" : [...]
    }
  ]
}
```

```
{
  "name": "BRP",
  "data": [
    [<"ALL" | "NONE">, <"ALL" | "NONE">],
    {
      "name" : <T_constructor_name>,
      "data" : [...]
    },

    {
      "name" : <T_constructor_name>,
      "data" : [...]
    }
  ]
}
```

4.5 Concrete Evidence Constructor Schemas

Corresponds to Figure 2

Note: Values for fields that hold “bits” should be standard base64-encoded strings (representing arbirary binary data–hashes, nonces, signatures, etc.).

```
{
  "name": "U",
  "data": [
    < number >,
    [< string >],
    < number >,
    < string >,
    {
      "name": <Ev_constructor_name>,
      "data": [...]
    }
  ]
}
```

```
{
  "name": "K",
  "data": [
    < number >,
    [< string >],
    < number >,
    < number >,
    < string >,
    {
      "name": <Ev_constructor_name>,
      "data": [...]
    }
  ]
}
```

```
{
  "name": "G",
  "data": [
    < number >,
    {
      "name": <Ev_constructor_name>,
      "data": [...]
    },
    < string >
  ]
}
```

```
{
  "name": "H",
  "data": [
    < number >,
    < string >
  ]
}
```

```
{
  "name": "N",
  "data": [
    < number >,
    < string >,
    {
      "name": <Ev_constructor_name>,
      "data": [...]
    }
  ]
}
```

```
{
  "name": "SS",
  "data": [
    {
      "name": <Ev_constructor_name>,
      "data": [...]
    },
    {
      "name": <Ev_constructor_name>,
      "data": [...]
    }
  ]
}
```

```
{
  "name": "PP",
  "data": [
    {
      "name": <Ev_constructor_name>,
      "data": [...]
    },
    {
      "name": <Ev_constructor_name>,
      "data": [...]
    }
  ]
}
```