Rockwell Collins

The Universal Short Graph Language

Introducing the ⊔ language

Laurent Fournier – lfournie@rockwellcollins.com

[3qoJE]*

May 31, 2012

1 chapitre

blabla

```
__RE_U__ = r'''      # RegExp
    (?:              # Three basic tokens:
     ([{}])          # (1) Block
    |
     (?:\#[^\n]*)    # or (2) Line comment
    |                # or (3) NODE:
     (?=[^\s<\-=>])  # Not empty token
     (?:(\w{1,10})|) # Name
     (?::(\w)|)      # Type pre
     ((%s)(.+?)\5|\[([^\]]+)\]|\(([^)]+)\)|) # Content
     (\w|)           # Type post
     (?:\.(\w{1,20}|\*)|) # Port
    |                # or (4) ARC:
     ([<\-=>])       # Head
     (?:([^\W\d_])|) # Type pre
     ((%s)(.+?)\14|\[([^\]]+)\]|\(([^)]+)\)|) # Content
     (?:([^\W\d_])|) # Type post
     ([<\-=>])       # Tail
)''' % (__delimiter__, __delimiter__)
```

2 Parser

blabla

```
    def parse(self, x):
        "kernel⊔parser"
        for p in __MACRO__:
            x = re.sub(r'\b%s\b'%p, __MACRO__[p], x)
        nodes, arcs, = {}, []
        sak = [(None, None),] # for parent setting
        sgl, cli, stl = False, (), [[],] # for arc setting
        for m in re.compile(__RE_U__, re.X|re.S).finditer(x):
            if sak:
                if m.group(1) == '{': # open block
                    sak.append((None, None))
                    stl.append([])
                    sgl = False
                elif m.group(1) == '}': # close block
                    sak.pop()
```

*the first five characters of the base64 encoding of the SHA1 digest of the attached source files.

```
                    stl.pop()
                    if sak: sak[-1] = (None, None)
                    sgl = False
                elif m.group(11): # link
                    sak[-1] = (None, None)
                    cli, sgl = self.typeLabel(m.groups()), True
                else: # node
                    (nid, typ, sep, lab) = self.typeLabel(m.groups(), False)
                    por, prt = self.getport(typ, m.group(10)), sak[-2][0] if
                        len(sak)>1 else None
                    if not prt and len(stl)>1: stl[-2] = []
                    if sgl and stl[-1]:
                        arcs += self.addarc(stl[-1][-1], (nid, por), cli)
                    sak[-1], sgl = (nid, por), False
                    stl[-1].append((nid, por))
                    self.merge_attr(nid, nodes, prt, typ, sep, lab)
        return nodes, arcs
```