The Habanero CnC Framework: A Demonstration of CnC Unification

Nick Vrvilo, Rice University

The 7th Annual CnC Workshop

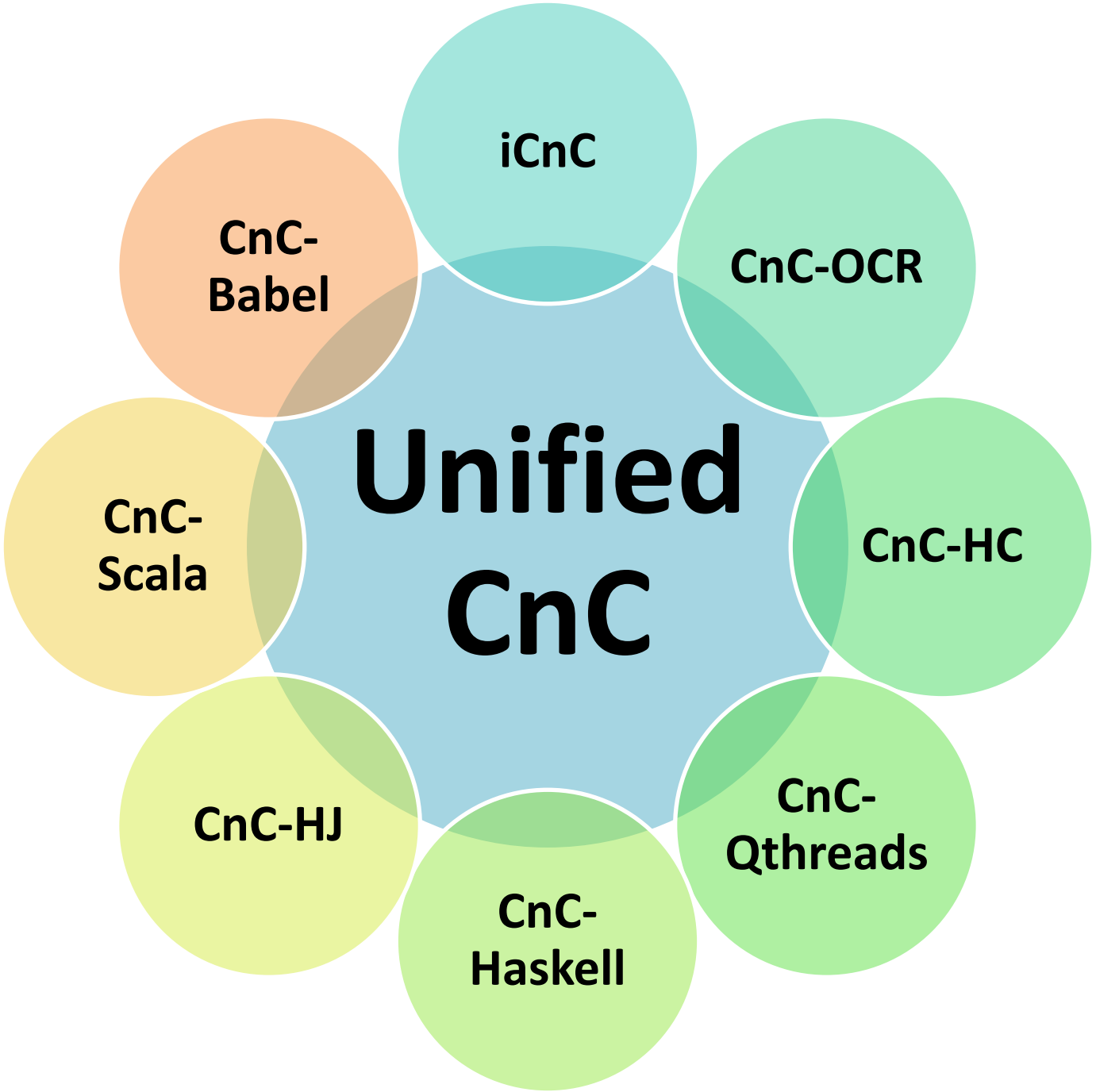
September 7, 2015

Acknowledgements

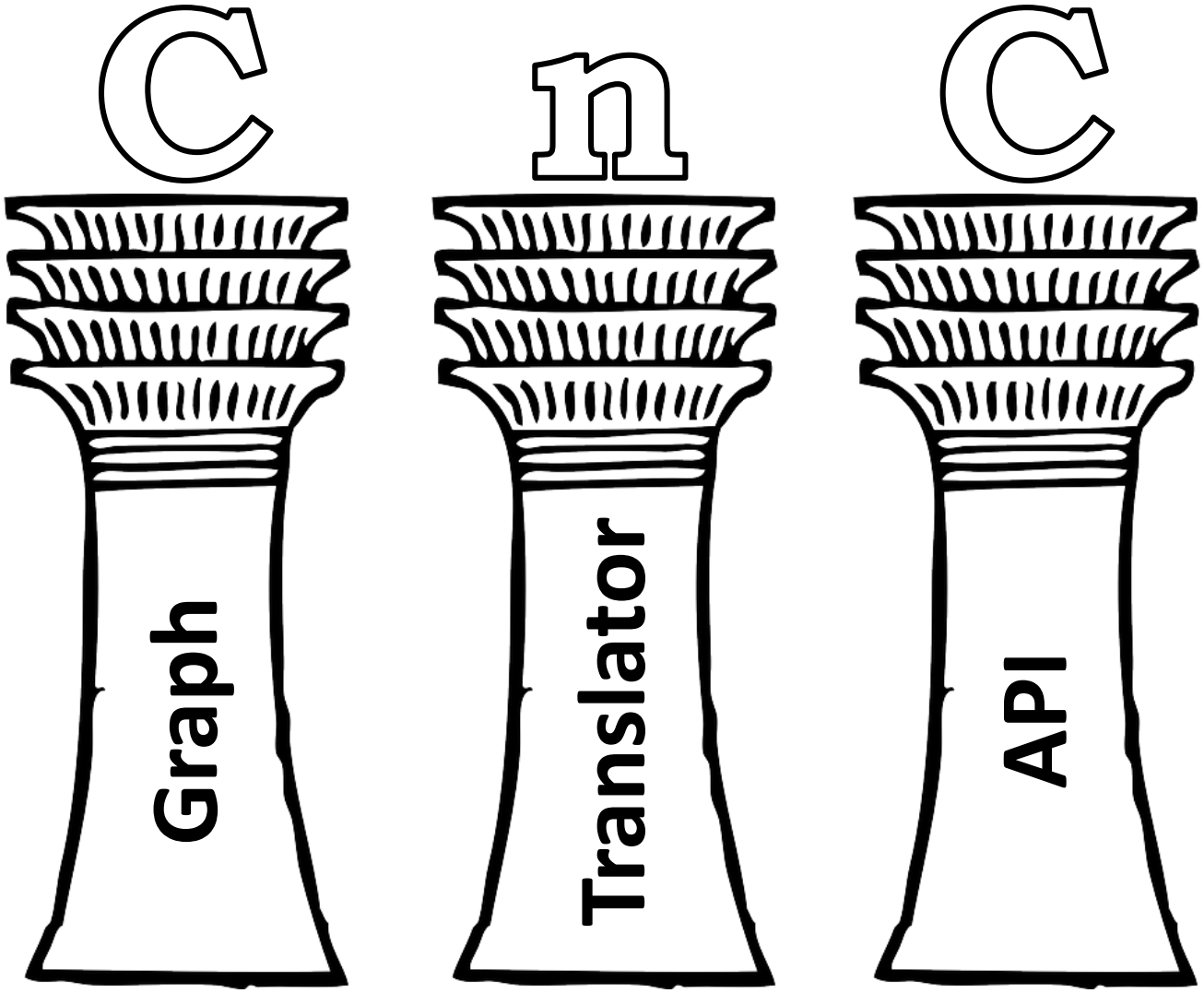
This work was inspired and influenced by the Open CnC community's unification discussions.

Frank Schlimbach, Kath Knobe, Zoran Budimlić, Alina Sbîrlea, Dragoş Sbîrlea, Gary Delp, and Ellen Porter.*

* I apologize if I left anyone out of this list!


iCnC
CnC-OCR
CnC-HC
CnC-Qthreads
CnC-Haskell
CnC-HJ
CnC-Scala
CnC-Babel
Unified CnC

Frank's 3 pillars of CnC unification



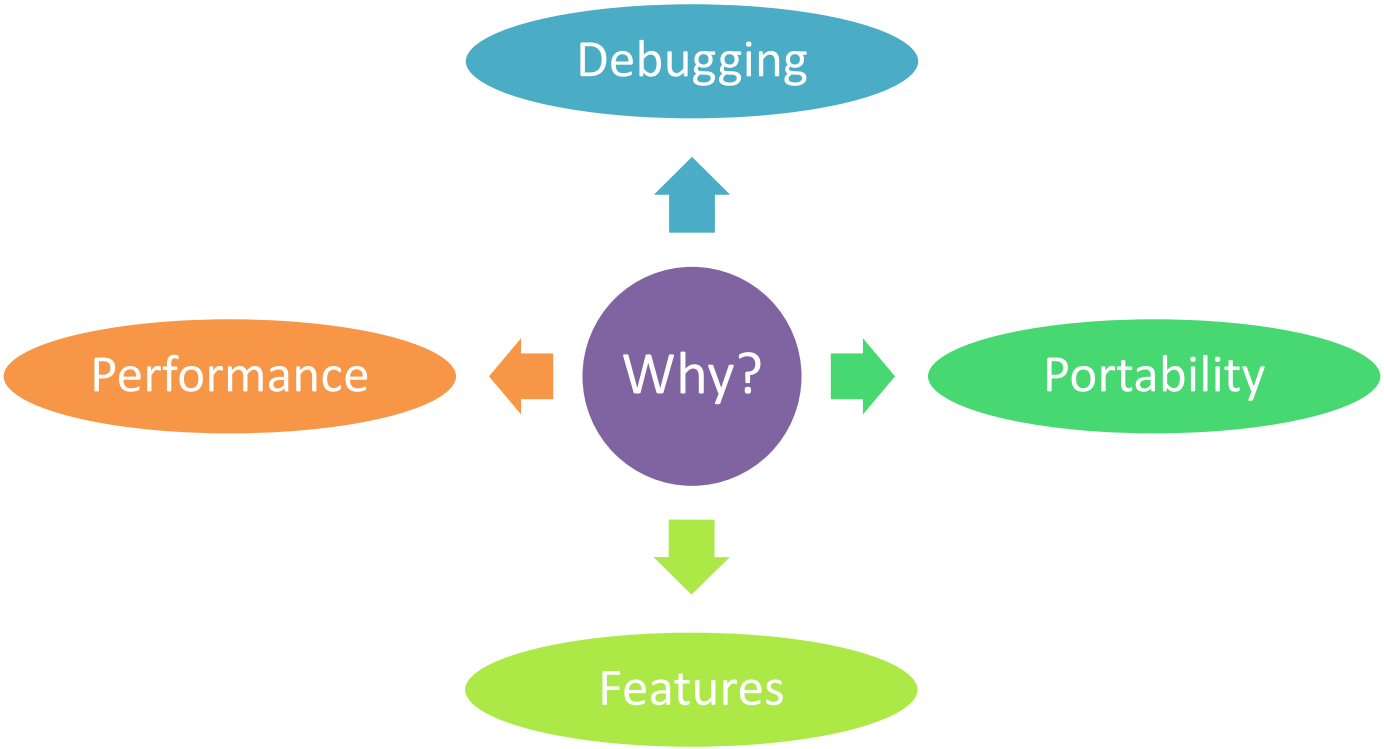
“Having a working demo does wonders for creating a consensus.”

Stephen Wong

(not an exact quote)

Nick's goal for unified CnC

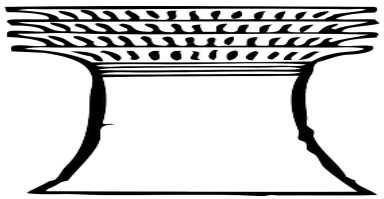
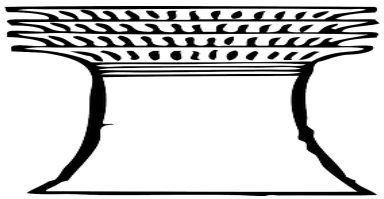
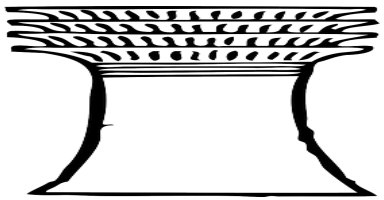
Run the same CnC app on both Intel CnC (MPI+TBB) and CnC-OCR



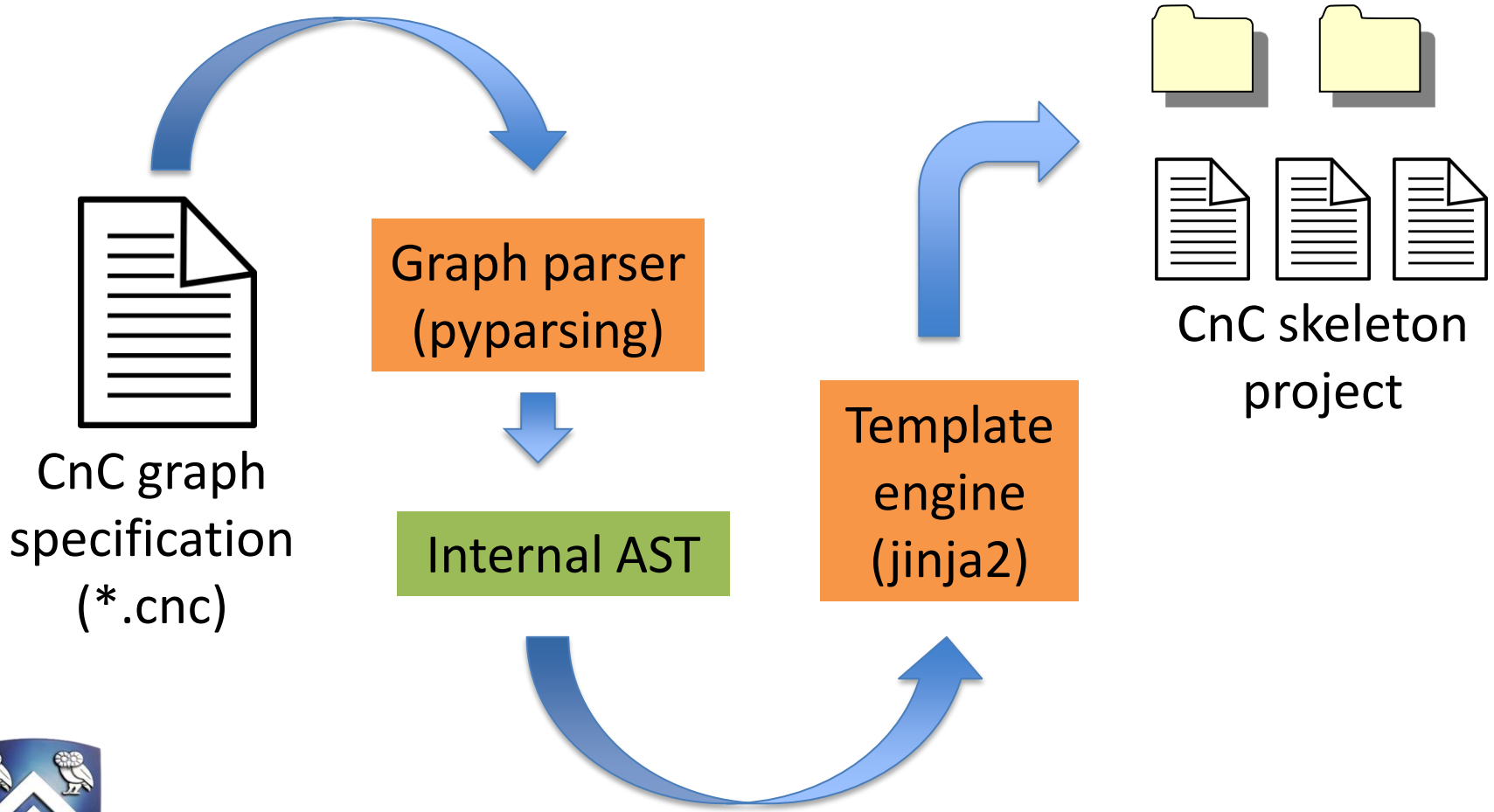
Nick's plan for unified CnC

Generalize the CnC-OCR Framework

- **Increase graph specification expressiveness**
- **Generate iCnC scaffolding**
- **Remove OCR abstractions from API**


Unified CnC translator design



Fibonacci's unified graph spec

```
$context { int n; };

[ int *fib: i ];

( $initialize: () ) -> ( compute_fib: $rangeTo(0,#n) );

( compute_fib: i )
 <- [ x @ fib: i-2 ] $when(i>1),
    [ y @ fib: i-1 ] $when(i>1)
 -> [ z @ fib: i   ];

( $finalize: () ) <- [ fib: #n ];
```

iCnC Fibonacci

```
int fib_step::execute( const int & tag, fib_context & ctxt )
   const {
    switch( tag ) {
        case 0 : ctxt.m_fibs.put( tag, 0 ); break;
        case 1 : ctxt.m_fibs.put( tag, 1 ); break;
        default :
            // get previous 2 results
            fib_type f_1; ctxt.m_fibs.get( tag - 1, f_1 );
            fib_type f_2; ctxt.m_fibs.get( tag - 2, f_2 );
            // put our result
            ctxt.m_fibs.put( tag, f_1 + f_2 );
    }
    return CnC::CNC_Success;
}
```

Fibonacci's unified graph spec

```
$context { int n; };

[ int *fib: i ];

( $initialize: () ) -> ( compute_fib: $rangeTo(0,#n) );

( compute_fib: i )
 <- [ x @ fib: i-2 ] $when(i>1),
    [ y @ fib: i-1 ] $when(i>1)
 -> [ z @ fib: i   ];

( $finalize: () ) <- [ fib: #n ];
```

Fibonacci's unified project

- cnc_support
  - x86
    - cncocr.h
    - cncocr.c
    - Fibonacci_internal.h
    - …
  - icnc
    - icnc.h
    - icnc.cpp
    - Fibonacci_internal.h
    - …
- Makefile.x86
- Makefile.icnc
- Makefile
- Fibonacci_defs.h
- Fibonacci.c
- Fibonacci_compute_fib.c
- Main.c
- build
- install

Fibonacci's unified step code

```
void Fibonacci_compute_fib(cncTag_t i,
        int *x, int *y, FibonacciCtx *ctx) {

    int *z = cncItemAlloc(sizeof(*z));
    *z = (i>1) ? *x + *y : i;
    cncPut_fib(z, n, ctx);

}
```

Unified CnC C API

```
int cncMain(int argc, char *argv[])

void G_S(cncTag_t t0, ..., cncTag_t tN, TX x, ..., TZ z, GCtx *ctx)
void G_cncInitialize(GArgs *args, GCtx *ctx)
void G_cncFinalize(cncTag_t t0, ..., cncTag_t tN, GCtx *ctx)

GCtx *G_create()
void G_destroy(GCtx *ctx)
void G_launch(GArgs *args, GCtx *ctx)
void G_await(cncTag_t t0, ..., cncTag_t tN, GCtx *ctx)

void cncPrescribe_S(cncTag_t t0, ..., cncTag_t tN, GCtx *ctx)

void *cncItemAlloc(size_t bytes)
void cncItemFree(void *item)
void cncPut_I(T *item, cncTag_t k0, ..., cncTag_t kN, GCtx *ctx)
```

Demo

https://youtu.be/R04HkX_97ww

Summary

- Realized 3 pillars of unification
  - Unified graph language
  - Unified graph translator toolchain
  - Unified developer API
- Proof of concept for iCnC and CnC-OCR
- Unified developer API is limited to C-compatible CnC runtime implementations

Future work

- Add additional back-end runtimes
  - CnC-HC via HC-Lib (+ MPI?)
- Add additional API targets
  - Intel CnC (C++ API)
  - JVM (HJ-Lib, Scala)
- Add JSON front-end support
  - Useful for JavaScript-based CnC designer

Go try it out!

https://github.com/habanero-rice/cnc-framework

https://xstack.exascale-tech.com/git/public/xstack.git

nick.vrvilo@rice.edu