CONCURRENT COLLECTIONS

Acknowledgements

- Slides and material from
  - Kathleen Knobe (Intel)
  - Vivek Sarkar (Rice University)

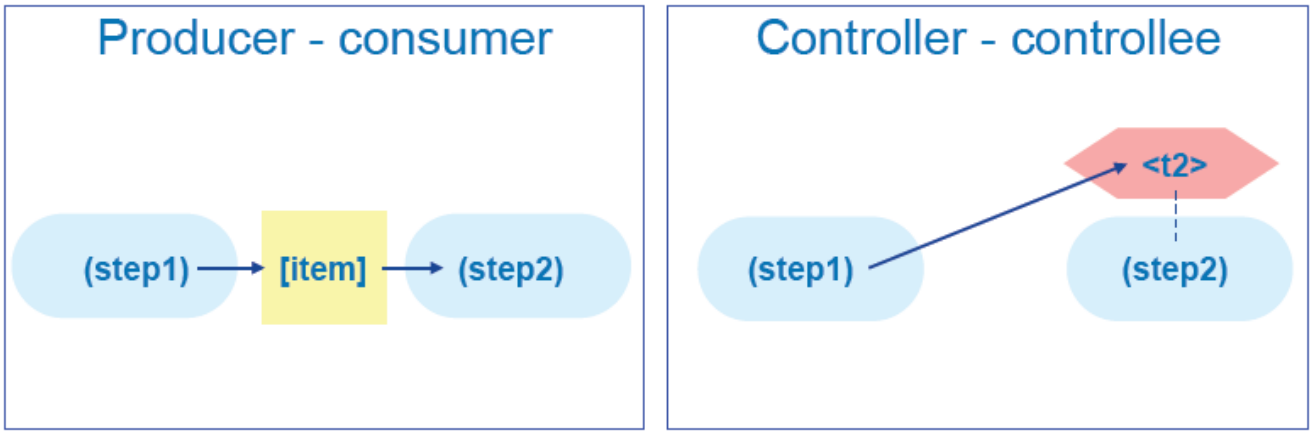
CnC – Concurrent Collections

- Declaratively specify the computation
- And the ordering constraints
  - Parallelism is implicit

- Available from Intel at
  - http://software.intel.com/en-us/articles/intel-concurrent-collections-for-cc/

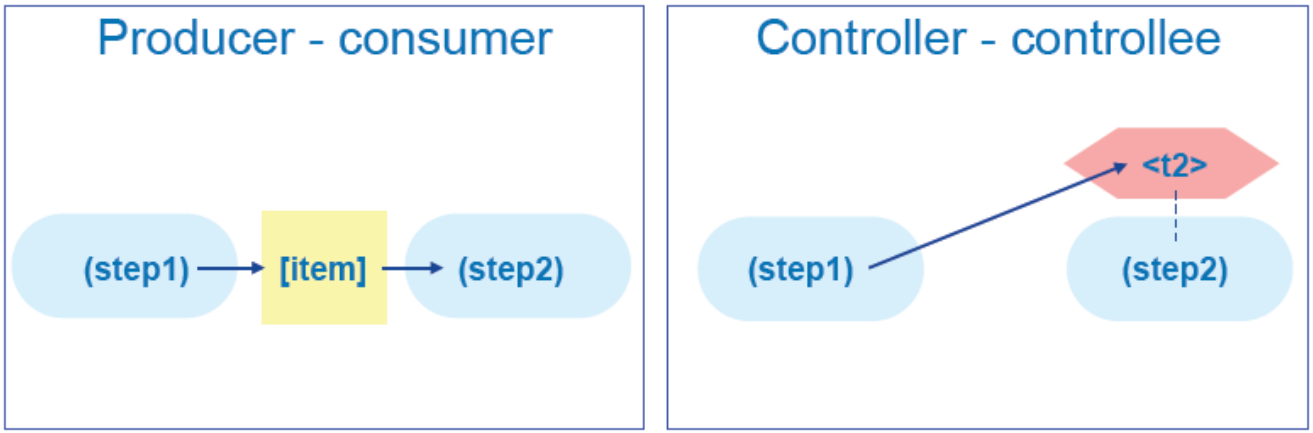
Two Sources of Dependencies

- Data Dependency (Producer Consumer)
  - Consumer of a data cannot execute before producer has produced the data
- Control Dependency
  - Controllee cannot execute unless its execution is determined as necessary by the Controller



Three Kinds of Collections

- (Step) – represents computation
- [Item] – represents data
- < Tag > – represents control flow



Dynamic Instances of Collections

- Multiple instances for each Step, Item, Tag collections at runtime
- Each instance has a unique Tag
- A tag is a tuple of values

Item Collection

- Each (data) item instance is dynamically written once
- Persists in the 'tuple space' forever
  - Garbage collection is implicit
- Each step instance can
  - Get() multiple item instances from multiple item collections
  - Put() multiple item instance into multiple item collections
- Each item can be Get() by multiple steps instance

Step Collection

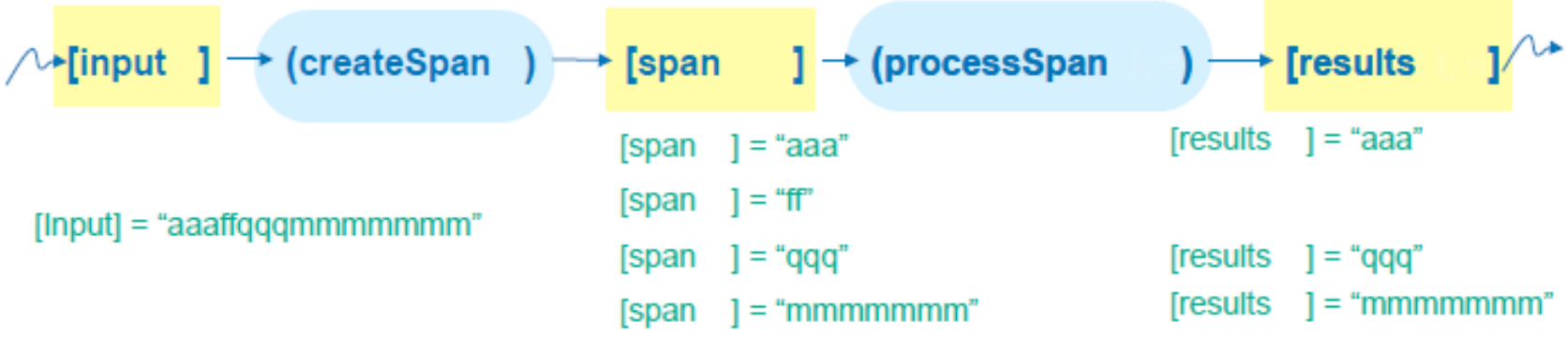
- Every step collection is “prescribed” by a Tag collection
  - An step instance with tag T can execute only if T is present in the prescribed Tab collection
- Each (computation) step instance is of the form
  - A sequence of Item Get()s
  - Determinstic side-effect-free sequential computation
    - A function of the tag and data items read
  - A sequence of Item Put()s and Tag Put()s

Tag Collection

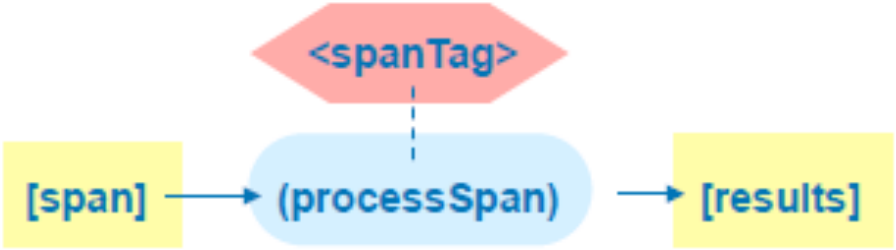
- Explicitly specifies the control flow between steps
- A tag collection might control multiple step collections

Simple Example

- Break an input string into sequences of repeated characters
- Filter only strings of odd length

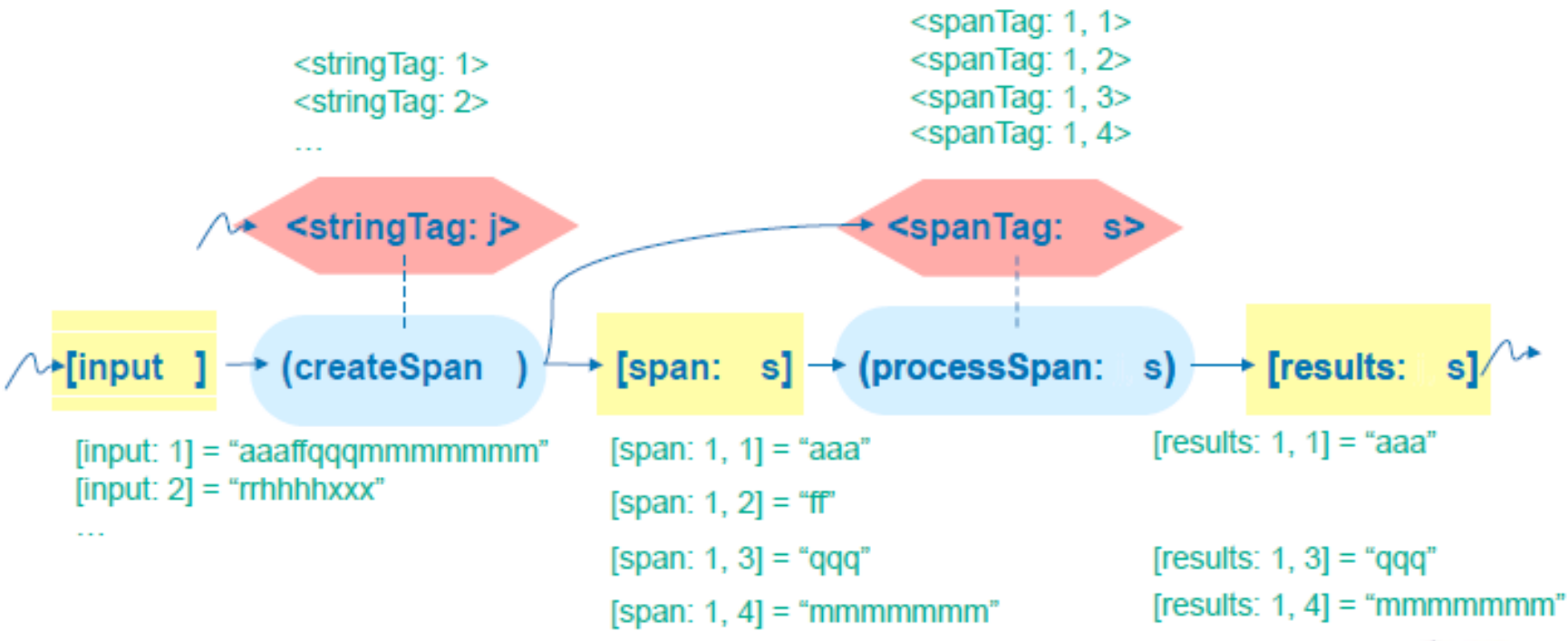


Execution Semantics



- A step instance becomes “proscribed” when its tag is available
- A step instance becomes “inputs available” when all its Get()s can succeed
- A step is enabled when it is “proscribed” and “inputs available”
- Enabled steps will eventually be executed

Control Tags Follow Data



Graph in Textual Form

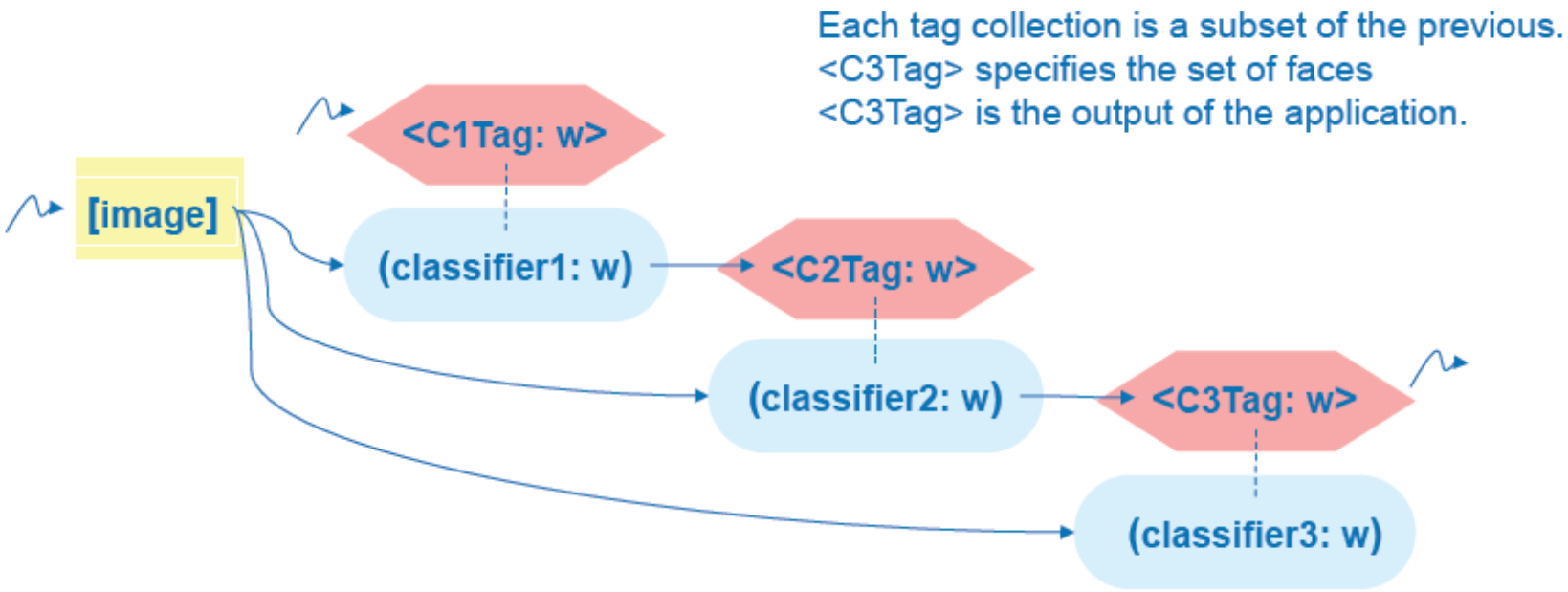
```
// control tags
<int singletonTag: singleton>;    <int spanTags: spanID>;

// data items
[string input <int>: singleton]; [string span <int>:
spanID];
[string results <int>: spanID];

// proscription relation
<singletonTag> :: (createSpan); <spanTags> :: (processSpan);

// program inputs and outputs
env -> [input]; env -> <singletonTag>; [results] -> env;

// producer/consumer relations
[input: singleton] -> (createSpan: singleton)
                   -> <spanTags: spanID>, [span: spanID];
[span] -> (processSpan) -> [results];
```

Coordination Separated from Code

```
int createSpan::execute(const int & t, partStr_context & c )
const {

    // Get input string
    string in;
    c.input.get(t, in);

    while (! in.empty()) {
        // construct span
        // …
        c.span.put(j, in.substr(j, len));
        c.spanTags.put(j);
        // …
    }

    return CnC::CNC_Success;
}
```

Control Separate From Data

- Face detection through a cascading sequence of classifiers



Scheduling Decisions

- When to execute enabled steps instances
- Where to execute enabled step instances
- Speculatively execute steps whose inputs are available
  - Based on availability of processor resources

Differences Between Linda and CnC

Is CnC Deterministic?

- For a given input, does CnC generate the same output?

Program Termination

- A CnC program terminates when
  - No step instance is executing
  - All unexecuted steps are not enabled

- Valid program termination
  - All proscribed steps are executed
  - i.e. A proscribed step still waiting for an input data ➔ error

Garbage Collection

- An item once produced logically stays forever
  - Can be garbage collected when no future step instance will issue a Get() with the same tag
  - A difficult analysis problem
- Current implementation
  - User provides a getCount() method
  - Determines the number of steps that will read the item