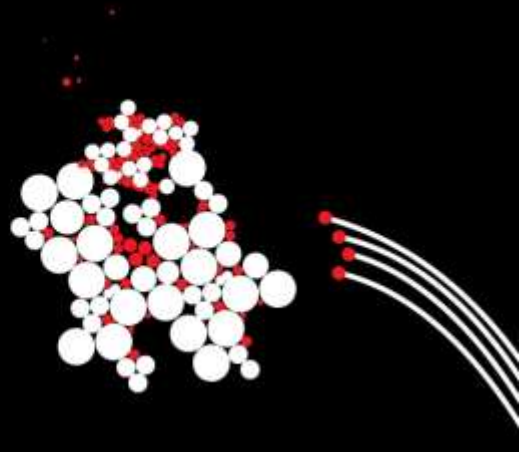
UNIVERSITY OF TWENTE.

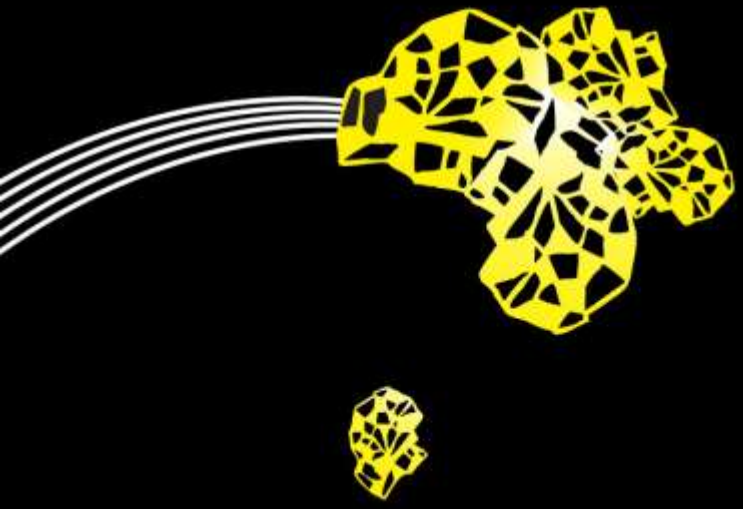


COMPILER CONSTRUCTION:

CC 6++: CC IN FORMAL METHODS AND TOOLS

5 JUNE 2019

MODULE 8: PROGRAMMING PARADIGMS



CC IN FORMAL METHODS AND TOOLS

- What is this?

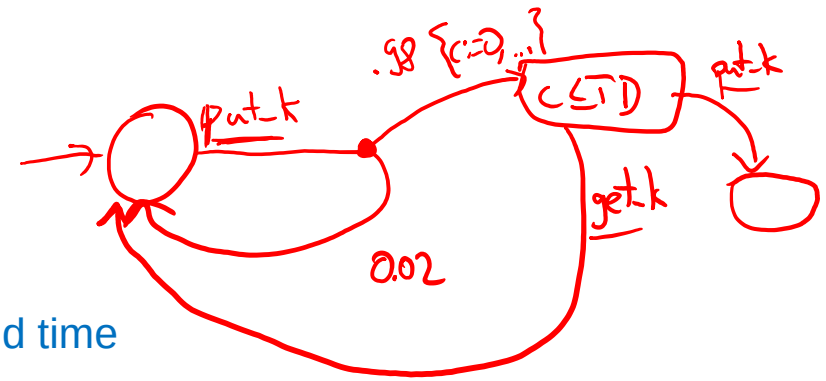
```
case '-':
  return new T() { Type = TokenType.Minus, Location = Span() };
case '*':
  return new T() { Type = TokenType.Multiply, Location = Span() };
case '/':
  c = PeekChar();
  if(c == '*') { ReadChar(); return SkipMultilineComment(isPropertyExpression); }
  else if(c == '/') { ReadChar(); return SkipSingleLineComment(isPropertyExpression); }
  return new T() { Type = TokenType.Divide, Location = Span() };
case '%':
  return new T() { Type = TokenType.Modulo, Location = Span() };
case '!':
  if(PeekChar() == '=') { ReadChar(); return new T() { Type = TokenType.NotEqual, Location = Span() }; }
  return new T() { Type = TokenType.Not, Location = Span() };
case (char)0:
  return new T() { Type = TokenType.EOF, Location = PointSpan() };
case 'a':
  if(PeekString("ny") && !IsIdentifierChar(PeekChar(2))) { ReadChar(1); return new T() { Type = TokenType.Nondete
  if(PeekString("rray") && !IsIdentifierChar(PeekChar(4))) { ReadChar(3); return new T() { Type = TokenType.Array
  goto default;
case 'd':
  if(PeekString("er") && !IsIdentifierChar(PeekChar(2))) { ReadChar(1); return new T() { Type = TokenType.Derivat
  else if(isPropertyExpression && PeekString("eadlock") && !IsIdentifierChar(PeekChar(7))) { ReadChar(6); return
  goto default;
```

  - part of a hand-written scanner
  - for the probabilistic modelling language Modest
  - from the code of the Modest Toolset: www.modestchecker.net

THE MODEST TOOLSET

- A set of tools for quantitative modelling and verification
- Modelling: with the Modest language

```
process ChannelK() {
    clock c;
    put_k palt  {
    :98: {= c = 0, inTransitK = true =};
         constrain(c <= TD) alt {
         :: get_k {= inTransitK = false =}
         :: put_k {= channel_k_overflow = true =}; stop
         }
    : 2: {==}
    };
    ChannelK()
}
```



- encodes a model with probabilities and time
- Markov chains, Markov decision processes (MDP), timed automata, …

CC IN FORMAL METHODS AND TOOLS

- What is this?

int[] a = [i+2, 4, j+4+2, 0];

```
// while(array.Length != i) {
var loopStartLabel = ilgen.DefineLabel();
var loopEndLabel = ilgen.DefineLabel();
ilgen.MarkLabel(loopStartLabel);
ilgen.Emit(OpCodes.Dup);
ilgen.Emit(OpCodes.Ldlen);
ilgen.Emit(OpCodes.Ldloc, i);
ilgen.Emit(OpCodes.Beq, loopEndLabel);

// array[i] = ...
ilgen.Emit(OpCodes.Dup);
ilgen.Emit(OpCodes.Ldloc, i);
var currentElementType = Dispatch(element.Expression, userState);
if(!LosslessCast(currentElementType, elementType, userState)) ErrorUnexpectedType(element.Expression);
ilgen.Emit(OpCodes.Stelem, elementType);

// ++i;
ilgen.Emit(OpCodes.Ldloc, i);
ilgen.Emit(OpCodes.Ldc_I4_1);
ilgen.Emit(OpCodes.Add_Ovf);
ilgen.Emit(OpCodes.Stloc, i);

// }
ilgen.Emit(OpCodes.Br, loopStartLabel);
ilgen.MarkLabel(loopEndLabel);
```

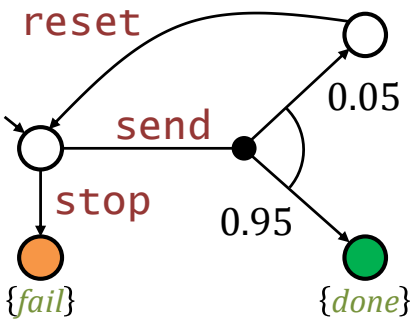
- part of a code generator, emits .NET bytecode for Modest models

VERIFICATION: MODEL CHECKING

formal model

```
process P() {
 alt {
 :: stop {= fail = true =}
 :: send palt {
    :95: {= done = true =}
    : 5: reset; P()
} } }
```

state space



exact results

$$\mathbb{P}_{\min}(\Diamond\, a) = 0.23$$
$$\mathbb{P}_{\max}(\Diamond\, a \wedge b) = 0.89$$
$$\mathbb{E}_{\min}(\#\mathtt{s} \mid b) = 12.5$$

state space exploration — ~~exploration~~ **explosion**

numeric computation — **issues w.r.t. convergence**

Markov decision process model checking

nondeterminism: abstraction, concurrency, …

probabilities: randomised algorithms, uncertainty, …

\+ guaranteed worst- and best-case results

CC IN FORMAL METHODS AND TOOLS

- What is this?

```
usedVariablesInfo = VariableInfoCollector.GetUsedVariablesInfo(network, out Changed);
foreach(var kvp in usedVariablesInfo)
{
  var sym = kvp.Key;
  var info = kvp.Value;
  var type = sym.Type;
  if(!ClockType.Instance.Equals(sym.Type) && !ContinuousType.Instance.Equals(sym.Type))
  {
    if(info.LowerBound != null && info.LowerBound.Equals(info.UpperBound))
    {
      // Remove the variable, replacing it by the only value ever assigned to it
      variableReplacement.Add(sym, null);
      variableReferenceReplacement.Add(sym, info.LowerBound);
      Changed = true;
    }
    else if(RealType.Instance.IsAssignableFrom(type) && !info.IsUsedAsReal)
    {
      // Change the type of the variable from real/int to int/bounded int
      // 1. Convert real to int
      if(!IntegerType.Instance.IsAssignableFrom(type)) type = IntegerType.Instance; // note: int is assignable from bounded

      // 2. Convert int to bounded int
      if(info.LowerBound != null && info.UpperBound != null)
      {
        if(type is BoundedType) continue; // do not change existing bounded ints
        type = new BoundedType(IntegerType.Instance, info.LowerBound, info.UpperBound);
      }

      // Create replacement variable
      if(type == sym.Type) continue;
      variableReplacement.Add(sym, new VariableSymbol(sym.Name, type, sym.IsTransient, sym.Location, sym.Comment));
      Changed = true;
    }
  }
}
```

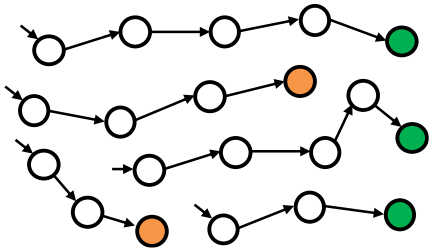
- an optimisation: tries to remove variables or make their type more specific

VERIFICATION: (DISCRETE-EVENT) SIMULATION

formal model

```
process P() {
 alt {
 :: stop {= fail = true =}
 :: send palt {
    :95: {= done = true =}
    : 5: reset; P()
} } }
```

sample runs



approx. results

$$\mathbb{P}_{\min}(\diamond\, a) \approx 0.2$$
$$\mathbb{P}_{\max}(\diamond\, a \wedge b) \approx 0.9$$
$$\mathbb{E}_{\min}(\#\mathtt{s} \mid b) \approx 12$$

simulation

statistics

…confidence intervals, Okamoto bound, sequential tests…

error bounds: e.g. result is ε-correct with probability δ

\+ constant memory usage, easy to parallelise

− rare events cause runtime explosion, requires a fully stochastic model

CC IN FORMAL METHODS AND TOOLS

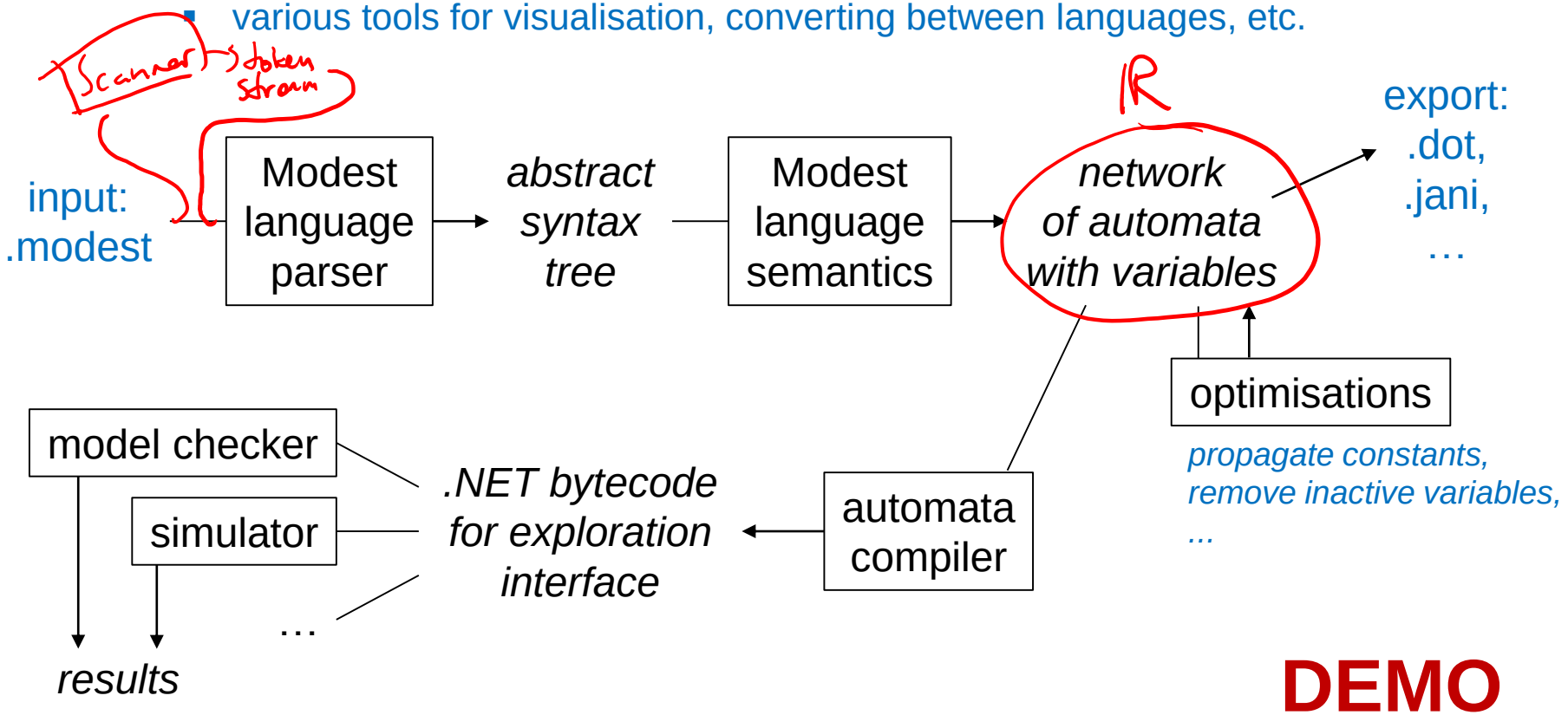
- What is this?

```
// TRY # LCURLY Behaviour RCURLY CatchBlock*
EatToken(TokenType.LeftCurly, state, ref loc);
var behaviour = ParseBehaviour(state);
EatToken(TokenType.RightCurly, state, ref loc);
var caughtExceptions = new List<ExceptionSymbol>();
var catchBlocks = new List<Construct>();
while(state.Current.ModestType == ModestTokenType.Catch)
{
  // CATCH ( IDENTIFIER | LBRACE IDENTIFIER RBRACE ) LCURLY Behaviour RCURLY
  state.Next();
  bool isBraced = state.Current.Type == TokenType.LeftBrace;
  if(isBraced) state.Next();
  string name = state.Current.Text;
  var excpLoc = state.Current.Location;
  EatToken(TokenType.Identifier, state, ref loc);
  if(isBraced) EatToken(TokenType.RightBrace, state, ref loc);
  caughtExceptions.Add(GetDummyException(state, name, excpLoc));
  EatToken(TokenType.LeftCurly, state, ref loc);
  catchBlocks.Add(ParseBehaviour(state));
  EatToken(TokenType.RightCurly, state, ref loc);
}
return new TryCatch(behaviour, caughtExceptions.ToArray(), catchBlocks.ToArray(), loc);
```

  - part of a hand-written recursive descent parser
  - for Modest, of course

THE MODEST TOOLSET

- Set of tools:
  - mcsta: probabilistic model checker
  - modes: discrete-event simulator (“statistical model checker”)
  - various tools for visualisation, converting between languages, etc.



DEMO