Collaborative Software Design & Development

Architecture

Dewayne E Perry
ENS 623A
Office Hours: T/Th 10:00-11:00
perry @ ece.utexas.edu
www.ece.utexas.edu/~perry/education/382V-s08/

Introduction

- 70s focus: software design
- 80s focus: integration of design aspects into programming languages
- 80s advance: description and analysis advances (eg, notation and typing) enable us to reason more effectively
- 90s: the decade of software architecture
- 00s: the decade of product line architectures, frameworks, COTS

Developing an Intuition

- Hardware Architecture
  - Multi-processor, pipe-lined, RISC
  - Interesting architectural points
    - Relatively small number of pieces
    - Scale: replication of components
  - BUT in software architecture
    - Exceedingly large number of components
    - Scale: not by replication but by addition of distinct pieces
  - Similarities, but fundamental differences
- Network Architecture
  - Star, ring, manhattan street networks
  - Interesting architectural points:
    - 2 components: nodes and interconnections
    - Small number of topologies
  - BUT in software architecture
    - Can abstract to this level
    - Large variety of topologies
    - Few named topologies
  - Do talk of distributed/message-passing architectures

Developing an Intuition

- Building architecture: *interesting architectural points*:
  - Multiple views
    - Elevation, floor plans
    - Scale models, structural, etc
  - Architectural styles
    - Specify constraints on design elements
    - Specify constraints on formal relationships
  - Relationship between architectural style and engineering principles
    - You don't get perpendicular style from romanesque engineering
  - Relationship between architectural style and building materials
    - You don't get skyscrapers from wooden post and beam construction
- Insights
  - Multiple views for insight and understanding
  - Styles as a cogent form of codification
  - Engineering principles are basic
  - Material properties are basic

Context: Where Does Architecture Fit?

- Requirements:
  - Information and processing
  - Characteristics of information/processing
- Architecture:
  - Elements and interactions
  - Constraints on elements/interfaces (properties, relationships)
- Design:
  - Modularization and interfaces
  - Algorithms/procedures and types
- Implementation:
  - Algorithm/procedure representations
  - Type representations

Caveats

- Different evaluations at different levels
- More of a continuum of refinement
- Requirements not so pure in practice; often contain
  - Architectural constraints
  - Design constraints
  - Implementation constraints

Motivation

- Cost factors in software architecture
  - Evolution
  - Customization
- Two architectural problems due to evolution
  - Architectural erosion: violation of architecture
  - Architectural drift: insensitivity to architecture
- Uses of SW Architecture
  - Prescribe constraints to desired level
    - Indicate restrictiveness/permissiveness
    - Define necessity and luxury
    - Pin-point relativeness and absoluteness
    - Ie, support principle of least constraints
  - Separate aesthetics from engineering
  - Express different aspects in appropriate manner
  - Perform dependency and consistency analysis

Standard Definitions & Our Model

- "the art or science of building: especially designing and building habital structures"
- "A unifying or coherent form or structure"

Software Architecture =

{

Elements,

Form,

Rationale

}

Our Model

- Three kinds of elements:
  - *Processing elements* supply transformations on data elements
  - *Data elements* contain the information that is used and transformed
  - *Connecting elements* are the "glue" that holds the various elements together – define the interactions

  [now often referred to as "components and connectors"]
    - A logical separation of processing/data and interaction
- Form: Consists of weighted properties and relationships
  - Weighting indicates load-bearing vs decoration
    - Indicates importance
    - Indicates alternatives
  - Properties constrain the choice of elements
    - What is not constrained is allowed
  - Relationships constrain the "placement"
    - How elements interact
    - How elements are organized

Our Model

- Rationale:
  - Justification of various aspects of the architecture
    - Economic considerations
    - Performance considerations
    - Reliability considerations
    - Functionality considerations
  - Makes explicit connections between aspects of the architecture and considerations
  - Make explicit interconnections with various aspects of requirements
  - The basis for performance analysis and simulation with respect to the various aspects of requirements

Architectural Style

- Abstracts elements and formal aspects
  - Possibly less complete
  - Possibly less constrained

  Eg, multi-process style, object oriented style
- A continuum – one's architecture may be another's style
- Importance of style
  - Encapsulates important decisions
  - Emphasizes important constraints
  - Coordinates multiple architects
  - Help prevent drift and erosion
- Styles may be global, regional, or local

Element Interdependence

- Important insight: multiple views
- Note the following observations:
  - A process view emphasizes data flow
  - A data view emphasizes process flow
  - A connector view emphasizes how various data and processing elements interconnect and interact
- All three views important and interdependent
  - Properties differentiate data states
  - Properties result from process transformations
  - Connectors must preserve or satisfy certain properties
- Want to move freely between the views

Example: Compiler Architecture

- Multi-phase Architectural Style
- Sequential Architecture
- Parallel Process, Shared Data Structure Architecture

- Multi-phase Architectural Style
  - *Processing elements:*
    - Lexer, Parser, Semantor, Optimizer, Code Generator
  - *Data Elements:*
    - Characters, Tokens, Phrases, Correlated Phrases, Annotated Phrases, Object Code
  - *Connecting Elements:*
    - (none specified)

- Some Data Element Relationships

Character/Token Relationship

- Processing view: Lexer: C -> T, where T preserves C
- Data View: Let C={c1, c2, …, cm} be a sequence of characters representing a source text, Cij I<=j be a subsequence of C whose elements are all the elements in C between ci and cj inclusive, T={t1, t2, …, tn} be a sequence of tokens, and indicate the correspondence between a token in T and a subsequence of C. T is said to *preserve* C if there exists an I, j, k, q, r and s such that 1<I<j<=m, 1<k<n, 1<q<=r<m , and for all t in T there exists a Cxy such that:

$$t \cong \begin{cases} C_i^1 & if\ t = t_1 \\ C_m^j & if\ t = t_n \\ C_r^q & if\ t = t_k\ where \quad \exists\, u, v \begin{vmatrix} 1 \le u \le q-1 \\ r+1 \le v \le m \\ t_{k-1} \cong C_{q-1}^u \\ t_{k+1} \cong C_v^{r+1} \end{vmatrix} \end{cases}$$

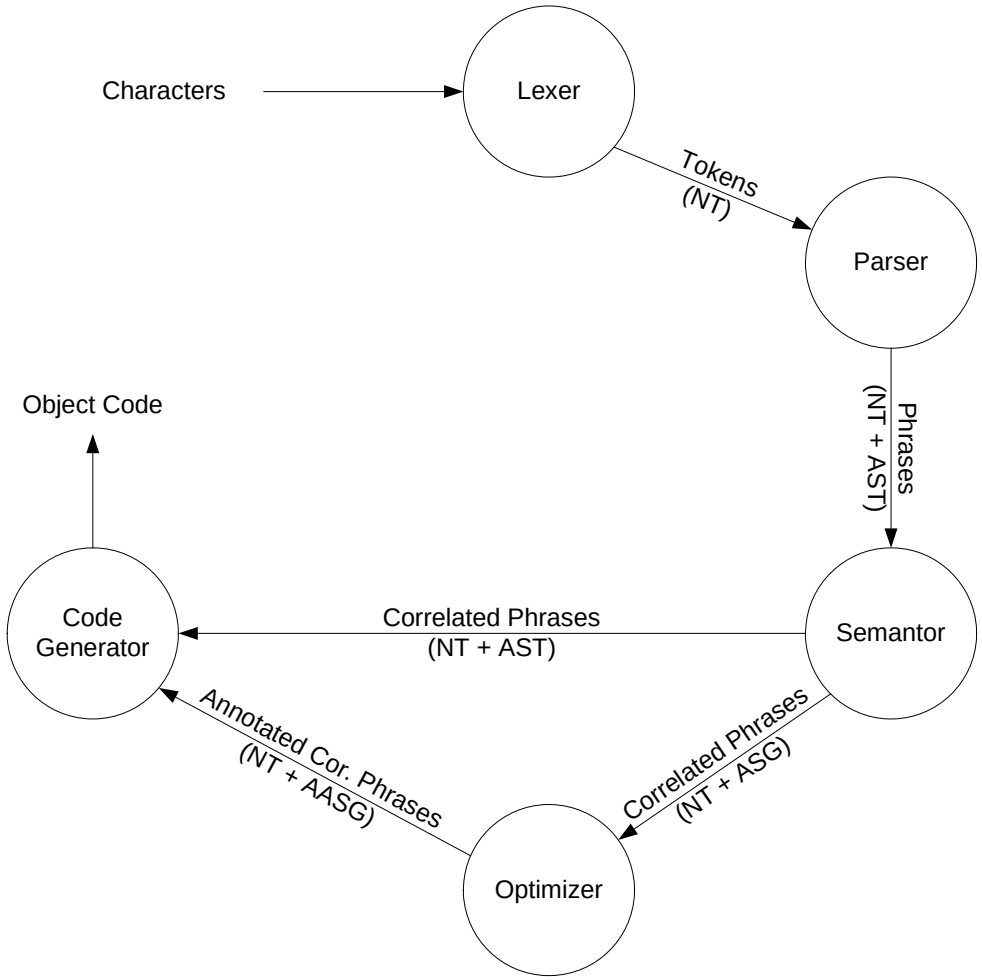
Lexer/Parser Relationship: Connector View

- Connectors must ensure that the tokens produced by the lexer are preserved for the parser, such that the order remains intact and that there are no losses, duplicates, or spurious additions

Sequential Compiler Architecture

- *Connectors*:
  - Procedure call and Parameters
- *Refine*:
  - Identifier tokens → Name Table (NT)
  - Phrases → Abstract Syntax Tree (AST)
  - Correlated phrases → Abstract Syntax Graph (ASG)
  - Annotated phrases → Annotated ASG

Partial Processing View



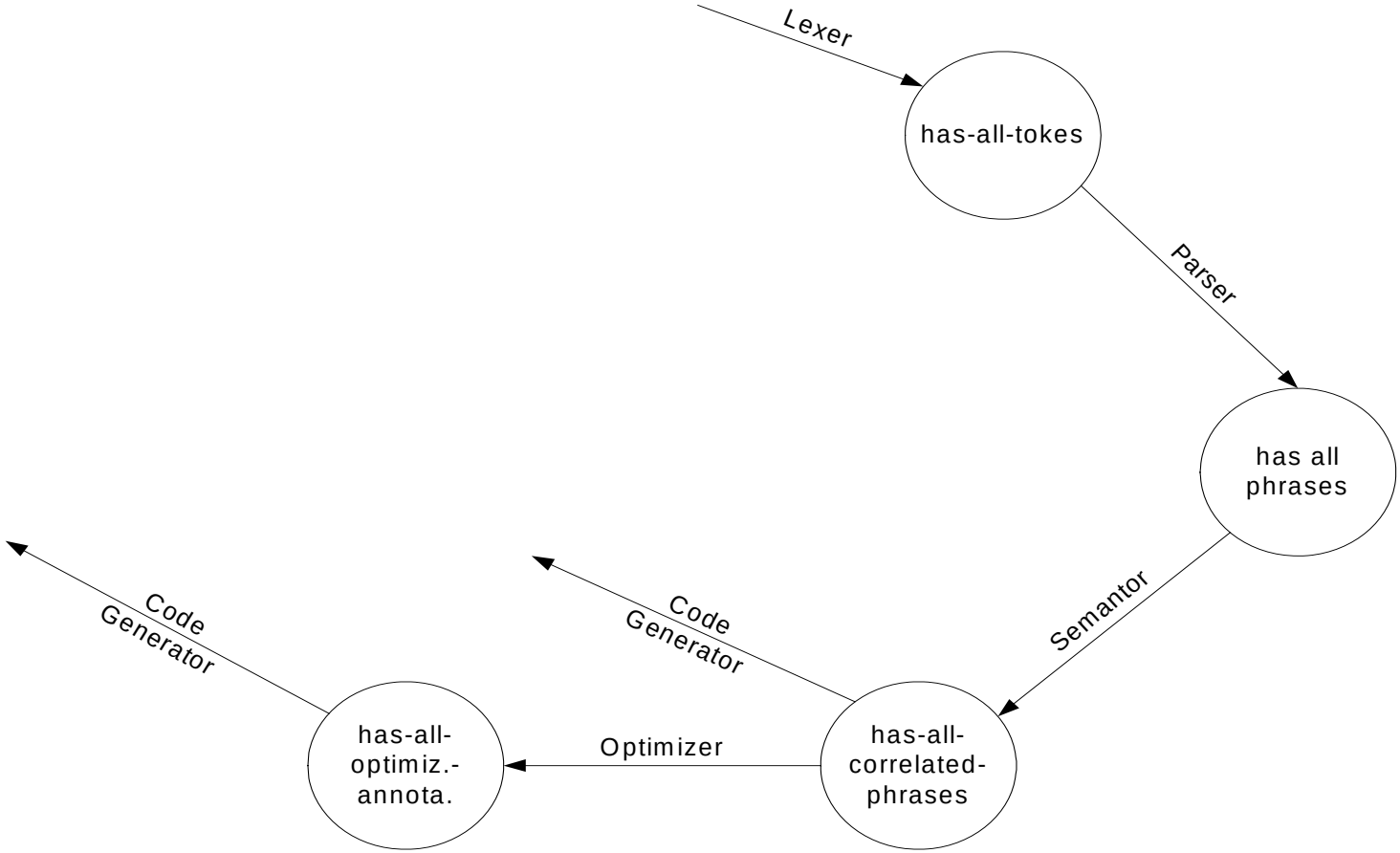
Application-Oriented Properties

- Describe data-structure states of interest to processing elements
- Examples:
  - Controlling processing order
  - Help define the effects of processing
  - Help define abstract operations needed by processing elements

Partial Data View

- *has-all-tokens*:
  - a state produced as a result of lexically analyzing the program text, necessary for the parser to begin processing
- *has-all-phrases*:
  - a state produced by the parser, necessary for the semantor to begin processing
- *has-all-correlated-phrases*:
  - a state produced by the semantor, necessary for the optimizer and code generator to begin processing
- *has-all-optimization-annotations*:
  - a state produced by the optimizer, preferred for the code generator to begin processing

Partial Data View



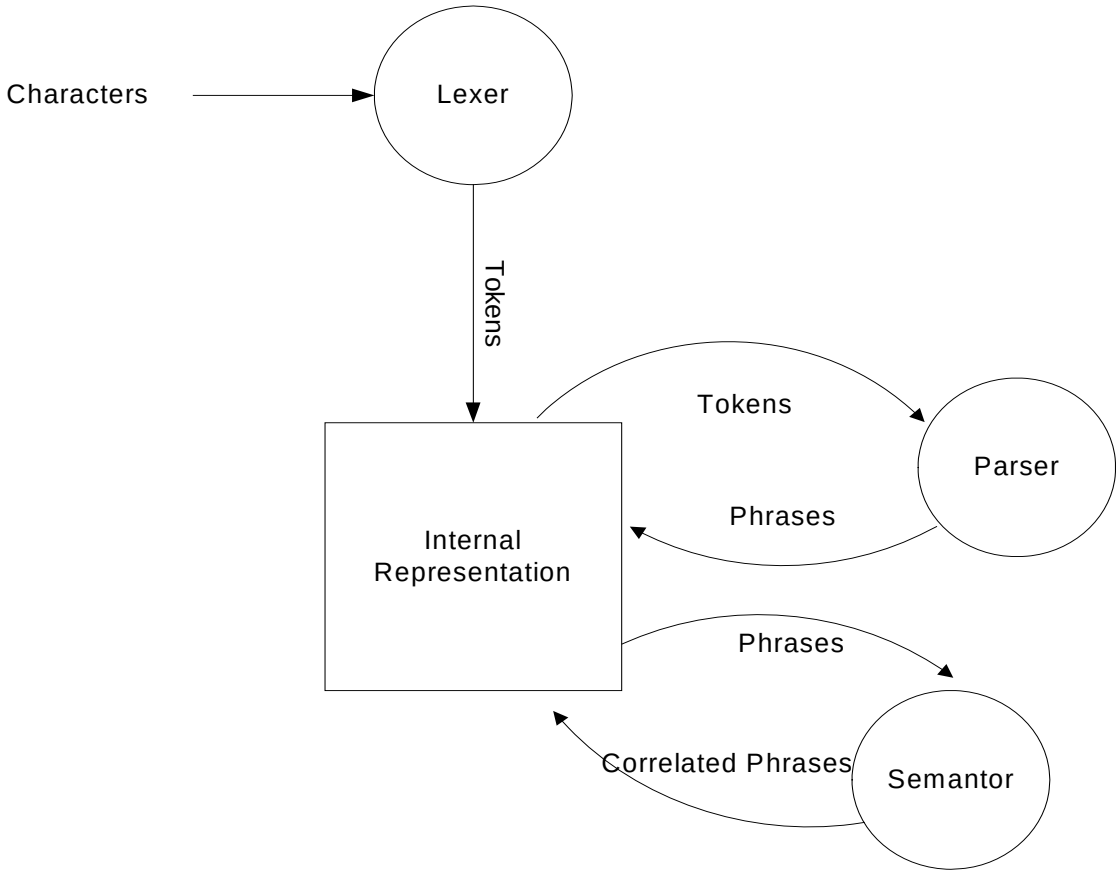
Sequential Compiler Architecture: Summary

- The form descriptions must include the relationships and constraints among the elements, including relative weightings and preferences
- Current type-based schemes for characterization elements are insufficient
- There is a natural interdependence between the processing and data views that can provide complementary descriptions of an architecture

Parallel Process, Shared Data Structure Architecture

- *Connecting Elements*:
  - Shared Representation
  - Parallel Execution with Eager Evaluation

Partial Processing View



Application-Oriented Properties

- >1 processing elements affecting state of representation
- Concurrent access to data structure
- Need coordination and synchronization

no-tokens	no-phrases	no-correlated-phrases
has-token	has-phrase	have-correlated-phrases
will-be-no-more-tokens	will-be-no-more-phrases	all-phrases-correlated

Connector/Data View

- Parallel Path Expressions for each data element
  (no-tokens, has-token+)*, will-be-no-more-tokens, has-token*, no-tokens
  (no-phrases, has-phrase+)*, will-be-no-more-phrases, has-phrase*, no-phrases
  no-correlated-phrases, (have-correlated-phrases)*, all-phrases-correlated
- Parallel Path Expressions relating data elements
  will-be-no-more-tokens, will be-no-more-phrases, all-phrases-correlated
  has-token+, has-phrase
  has-phrase+, has-correlated-phrase

Processing View

- Parallel Path Expressions for each processing element

  lexer: (no-tokens, has-token+)*, will-be-no-more-tokens

  parser: no-phrases, (has-token+, has-phrase)*, will-be-no-more-tokens, (has-token+, has-phrases)*, no-tokens, will-be-no-more-phrases

  semantor: no-correlated-phrases, (has-phrase+, has-correlated-phrases)*, will be-no-more-phrases, (has-phrase+, has-correlated-phrases)*, no-phrases, all-phrases-correlated

Relating Architectures

Sequential Arch	Parallel Arch
has-all tokens	will-be-no-more-tokens
has-all phrases	will-be-no-more-phrases
Has-all-correlated-phrases	all-phrases-correlated

Parallel Process, Shared Data Structure Compiler Architecture: Summary

- The processing elements are much the same as in the previous architecture, but with different "locus of control" properties
- The form of this architecture is more complex than that of the previous one --- there are more application-oriented properties and those properties require a richer notion to express them and their interrelationships
- We still benefit for the processing/data/connector view interdependence, albeit with more complexity
- Application-oriented properties are useful in relating similar architectures

Summary

- Separates out useful level of concern
  - problem domain meets implementation domain
- Defines important constraints on the system
- Basic structure of the system
- Means of capitalizing on assets
- Moves us from integral to compositional
- Integrates composition with generation

Perhaps the reason for such slow progress is that we have trained carpenters and contractors but no architects