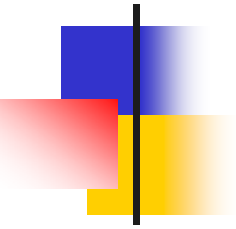
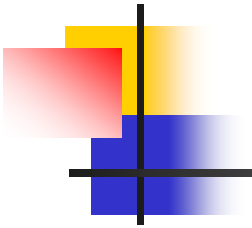


Design Pattern Detection Tools

Shouzheng Yang

Oct. 24, 2011



PINOT

- Motivation & Intent
  - GoF book categorizes the 23 patterns based on their purposes (creational, structural, or behavioral).
  - But for the scope of design pattern detection……

    Singleton (a creational pattern)

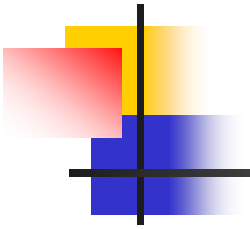
    Requires not only detecting the existence of object creation, but also verifying the behavior of the method body that creates and returns the Singleton instance.

    Flyweight (a structural pattern)

    Requires behavioral analysis to verify whether all flyweight objects in the flyweight pool are singletons and are created on demand.

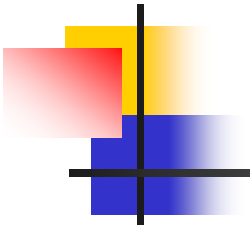
    Template Method, **Visitor** (both behavioral patterns)

    Define their behavior in the class definitions, which can be identified based on static structural analysis.



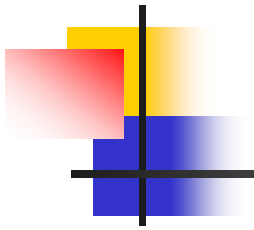
PINOT: Pattern Re-classification

- Language-provided Patterns (ignored)
  - Pattern components already provided in the language.
  - Developers tend to rely on programming languages' build-in facilities for the implementation.
  - E.g. Iterator (java.util.Iterator), Prototype (the clone() method in the Object class)



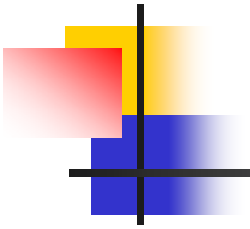
PINOT: Pattern Re-classification

- Structure-driven Patterns
  - Patterns that can be detected by inter-class relationships.
  - Inter class relationships typically include declaration, generalization, association, and delegation relationships.
  - E.g. Template Method, Composite, Bridge, Adapter etc.



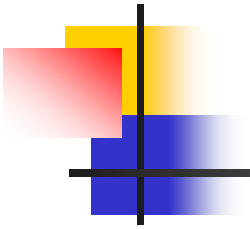
PINOT: Pattern Re-classification

- Behavior-driven Patterns
  - Patterns that are designed to realize certain behavioral requirements.
  - The pattern intent is carried in inter-class relationships and method bodies.
  - E.g. Strategy, State, Singleton, Chain of Responsibility, Decorator, Observer, Mediator etc.



PINOT: Pattern Re-classification

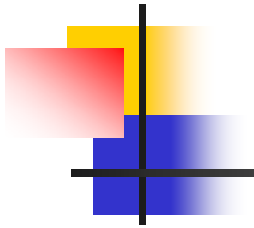
- Domain-specific Patterns (ignored)
  - Patterns that are specialized to suit a particular domain.
  - E.g. Interpreter, Command.
- Generic Concepts (ignored)
  - Patterns that are too generic to contain traceable implementation.
  - E.g. Memento, Builder.



PINOT: Tool Implementation

- Relies on a compiler - Jikes
  - Abstract syntax tree
- Data-flow analysis
  - Singleton (lazy instantiation)

    Analyze the flag variable that guards the lazy instantiation.
- Identifies array and array indexing, collection and its iteration
  - Observer



PINOT: Conclusion

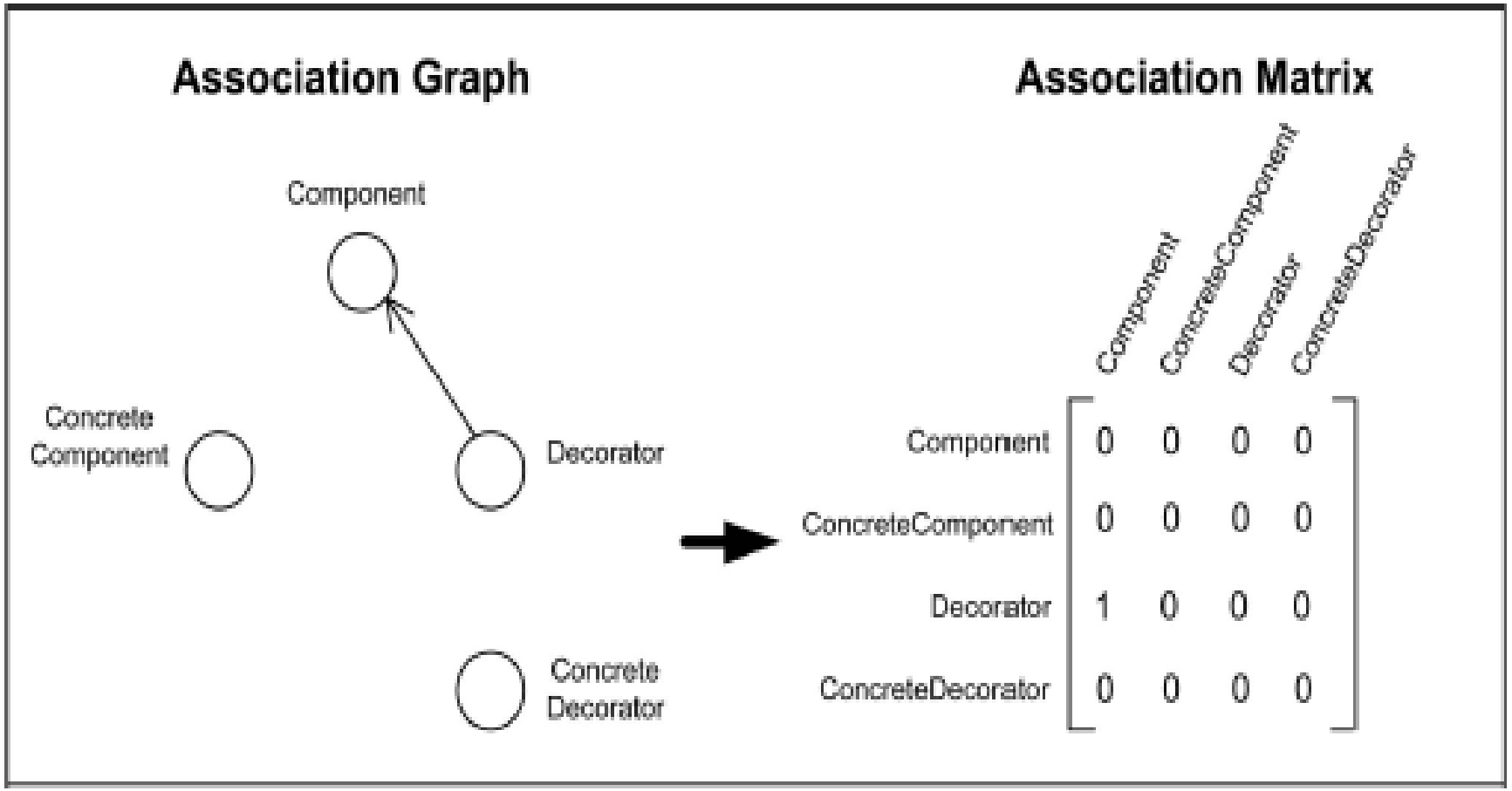
Treat patterns differently,

and deliberate on the detail!

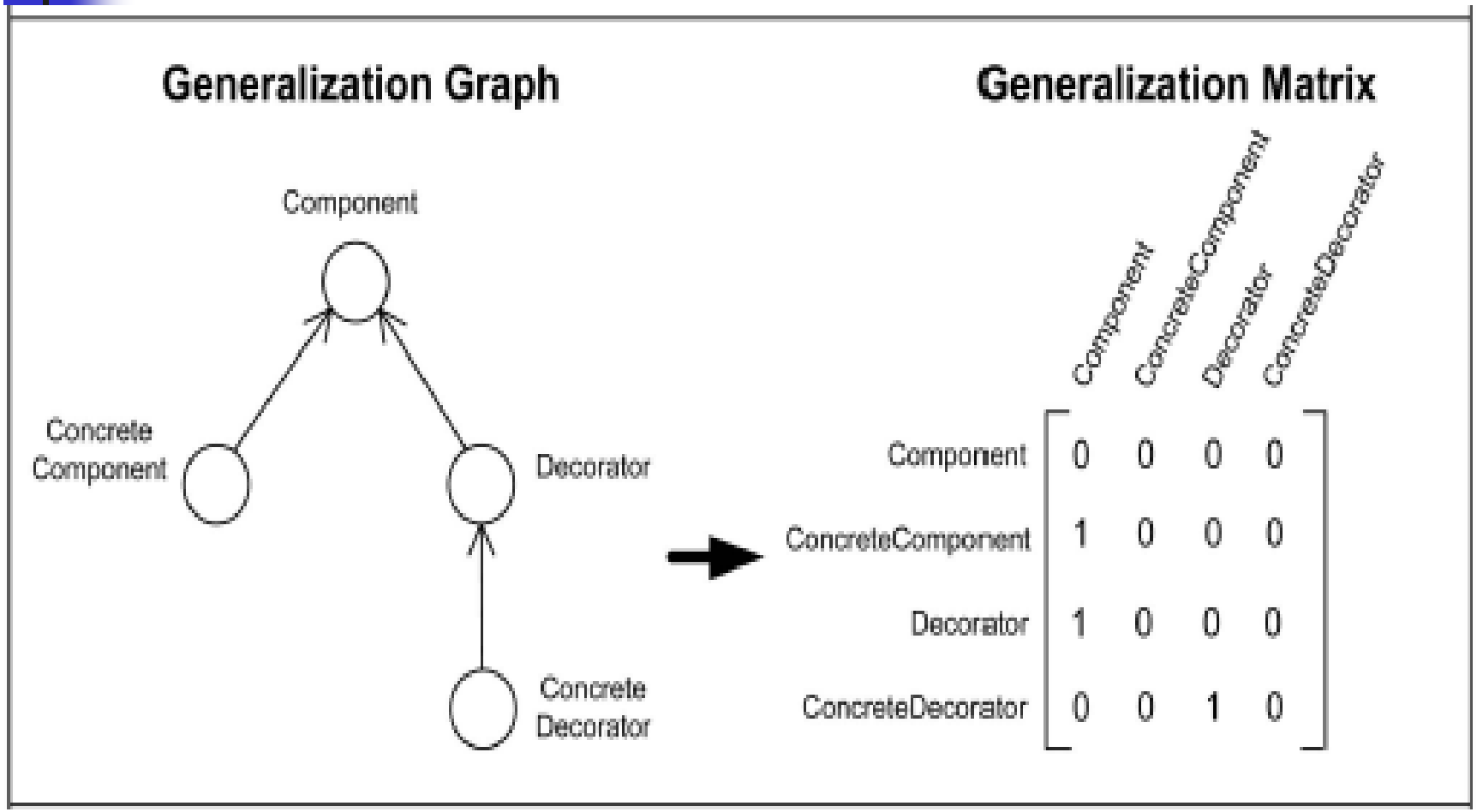
SSA: Similarity Scoring Algorithm

- Motivation
  - A detection methodology should not be based on specific patterns.
- General Idea
  - Treat system and pattern structures as directed graphs.
  - Find a way to represent these directed graphs.
  - Match the system's graph (sub-graph) representation with the actual pattern's representation.
  - Then ……

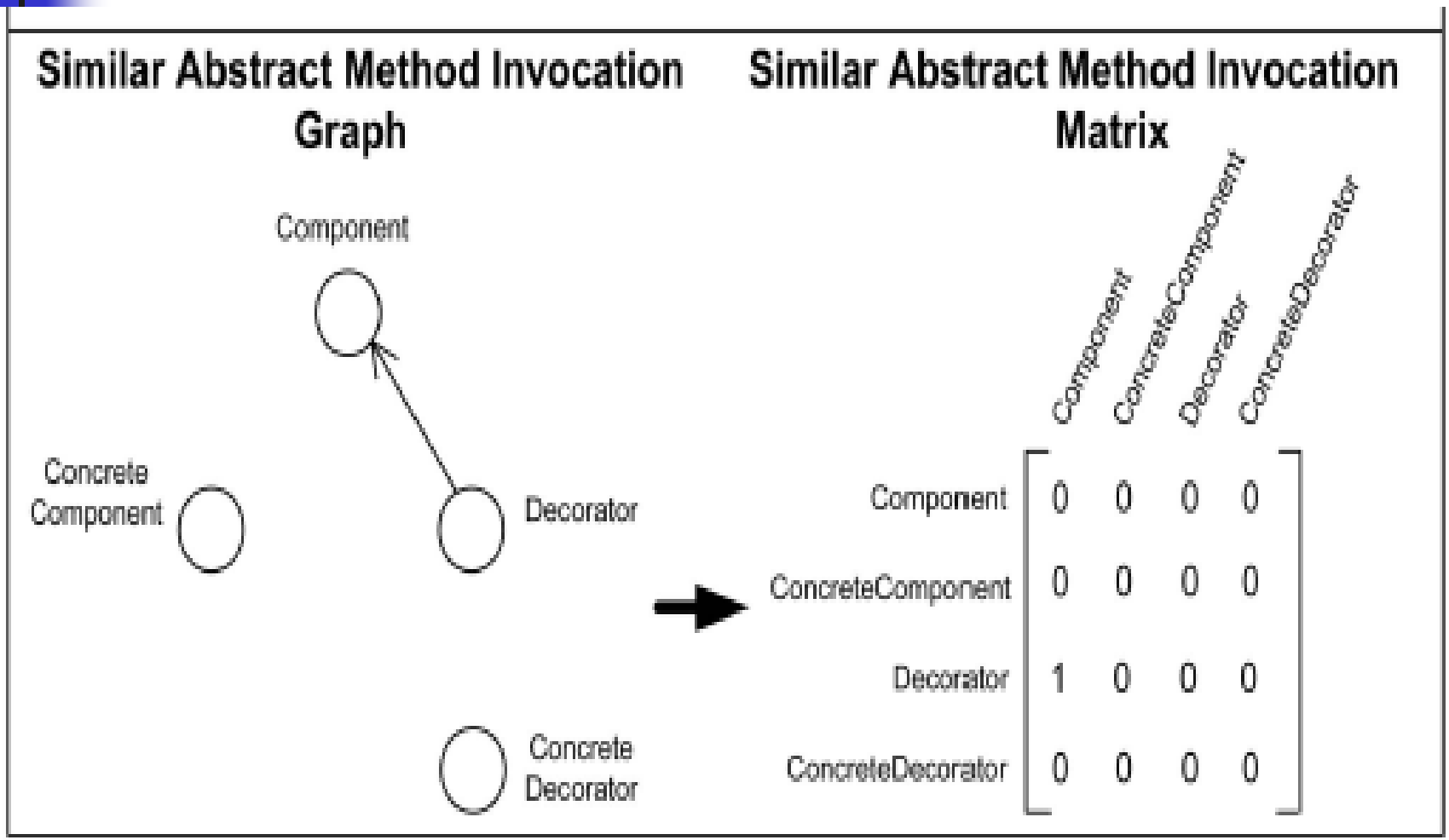
Representation of Systems and Patterns

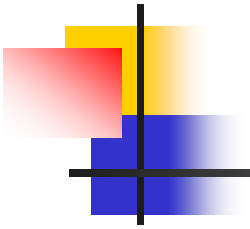


Representation of Systems and Patterns



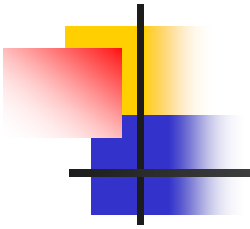
Representation of Systems and Patterns





Similarity Scoring Algorithm

- The algorithm
  - Quality of a web page
- SSA vs. Two general graph matching algs.
  - Exact matching (one-to-one mapping)
  - Inexact matching

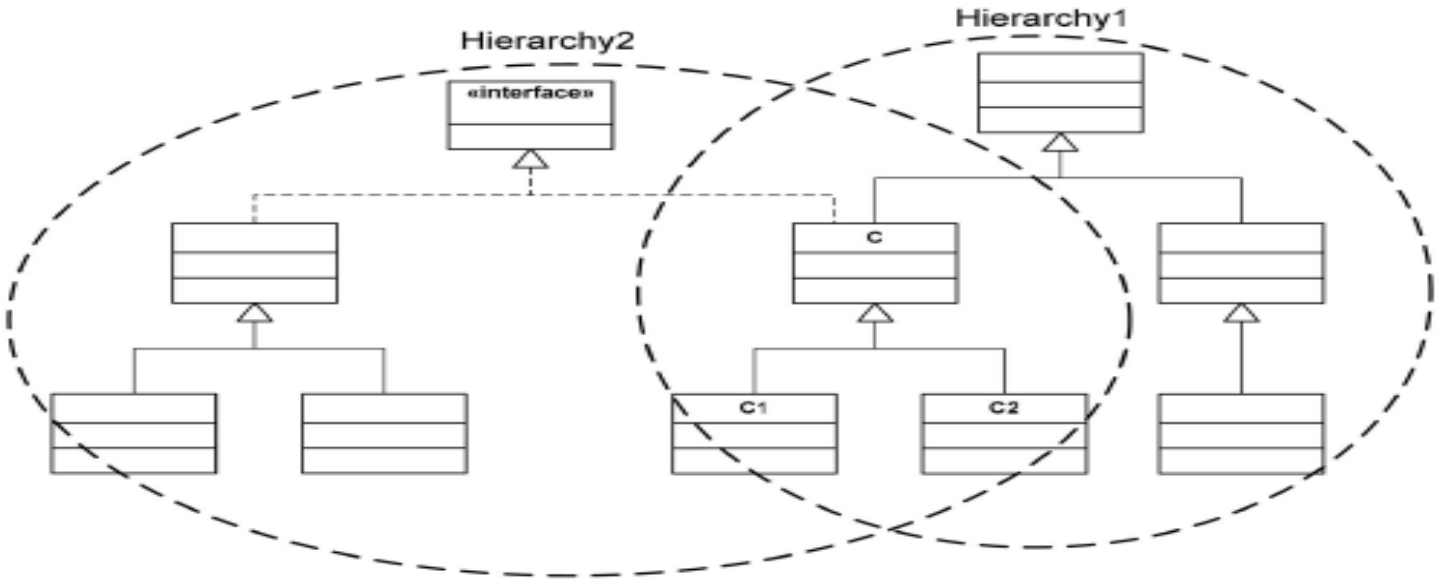


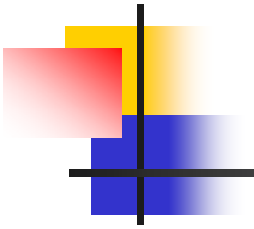
Similarity Scoring Algorithm

- Methodology:
  - Represent the system and patterns
  - Apply the algorithm formula for all sub-graphs
  - Extract the pattern instances
    - Sort scores.
    - Threshold is required.
  - But this is very slow!

Sub-systems

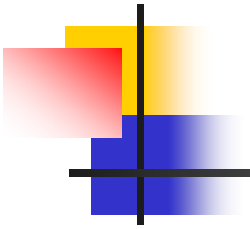
- Take advantage of the fact that most design patterns involve inheritance hierarchies.





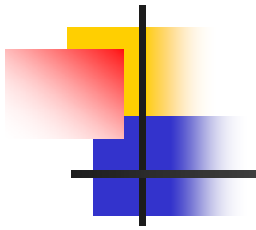
Sub-systems

- More steps in the methodology:
  - Representation of the system and patterns
  - Detection of inheritance hierarchies
  - Construction of subsystem matrices
    - One hierarchy. E.g., Composite, Decorator
    - More than one. E.g., Visitor, State
  - Applying the algorithm formula
  - Extraction of the pattern instances



Similarity Scoring Algorithm

- Optimization options
  - Assign weight to each matrix.
  - Exclude irrelevant subsystems.
- Implementation
  - Based on a Java byte-code manipulation framework.



Reference

- Tsantalis, N., Chatzigeorgiou, A., Stephanides, G., and Halkidis, S. T. (2006). Design Pattern Detection Using Similarity Scoring. IEEE Trans. Softw. Eng., 32(11):896-909.
- Shi, N. and Olsson, R. A. (2006). Reverse Engineering of Design Patterns from Java Source Code. In ASE '06: Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering, pages 123-134, Washington, DC, USA. IEEE Computer Society.