Equivalence in Calculi for Concurrency

1. The Join Calculus: a Language for Distributed Mobile Programming

```
@inproceedings{Fournet:2000,
  author = {Fournet, C{\'e}dric and Gonthier, Georges},
  title = {The Join Calculus: A Language for Distributed Mobile Programming},
  booktitle = {Applied Semantics, International Summer School},
  year = {2000},
  isbn = {3-540-44044-5},
  pages = {268--332},
  publisher = {Springer-Verlag},
}
```

Summary: Fournet and Gonthier describe the join calculus, from its motivations, to its semantics and equational theories. Influenced by the π calculus, and more generally earlier process calculi, the join calculus admits a wide range of identities. The majority of the notes compare and contrast the design trade-offs of different classes of equivalence to provide a hierarchy of equivalences, including may testing, must testing, bisimilarity, labeled bisimilarity, to name a few. In the final section, extensions of the join calculus are described to handle distribution and mobility.

Evaluation: Though many of the ideas described in these notes summarize past results, these notes provide a great introduction to the area of process calculi, providing motivation and intuition for the advances made -- and challenges faced -- in this area for the last 40 years.

2. Communication and Concurrency

```
@book{Milner:1989,
  author = {Milner, Robin},
  title = {Communication and Concurrency},
  year = {1989},
  isbn = {978-0131150072},
  publisher = {Prentice Hall}
}
```

Summary: This textbook describes the Calculus of Communicating Systems (CCS). Branching away from models of sequential computation such as the Lambda Calculus, CCS models concurrent computation where *processes* communicate with each other. Central to the language is parallel composition of processes, along with nondeterministic choice. With these primitive operations (and others) in CCS, the book describes how to model communicating systems and then spends the majority of the book discussing how to reason about their properties by providing a rich set of equational theories and proof techniques.

Evaluation: Like the previous publication but for the join calculus, this is not the first work describing CCS (for that, see Milner's previous textbook published in 1980). However, due to its updates (including an improved equational theory that provides coinductive definitions of bisimulation), this textbook strongly influences the future modeling and equational theory of process calculi (including the join calculus).

3. On the origins of bisimulation and coinduction

```
@article{Sangiorgi:2009,
  author = {Sangiorgi, Davide},
  title = {On the Origins of Bisimulation and Coinduction},
  journal = {Transactions on Programming Languages and Systems},
  volume = {31},
  number = {4},
  year = {2009},
  pages = {1--41},
  publisher = {ACM}
}
```

Summary: This paper describes the origins of bisimulations and coinductive definitions in Computer Science, logic, and set theory. It provides a self-contained background (through the use of an abstract labeled transition system) with definitions of bisimulation and bisimilarity and the fixed-point theorems required to coinductively define those equational theories. It also provides historical perspective for how Milner and Park reached the theories we use today in modern process calculi (such as the join and π calculi).

Evaluation: This paper focuses on what the authors of process calculi have considered their "power horse:" the coinductive definitions of equivalence. Specifically, it gives some context for Milner's first attempts at equivalence in CCS and how David Park provided insight into improvements to his definitions in order to invoke results from fixed-point theory (i.e., Tarski's fixed-point theorem). Though not directly relevant to process calculi, it is also interesting to see how other areas of math independently discovered these notions.