1 literature

1.1 Dictionary

(clarify the use of Agent, nodes, locations and be consistent throughout the thesis.) Literature uses different kinds of naming conventions for talking about communication paricipants, sometimes to express a slightly different intent that share semantically the same properties. I will stick to location throughout my work, in other work you might read terms like nodes, parties, Agents. The paper also presents several extensions (... maybe add whats important)

1.2 Sessions and Session Types: an Overview

Session Types take an important role for Choreos and describe the interaction of two (binary) or more (multi party) locations at communication level. We differentiate between **Global Session Types** that describe the sequence of directed message exchange for a session, and **Local Session Types**, which reflect the same Protocol but from one locations's perspective. The projection from a global session type to a local ST is called **End Point Projection** and basicallay translates all sending operations to send operations if location being projected to is sending, or receive operations if is receiving.

1.3 HasChor: Functional Choreographic Programming for All (Functional Pearl)

this paper is about Choreographic programming and an implementation for the Haskell Language. Choreographies are implemented as a Library for use with Haskell and expressed as Computational monads. Might be helpful for own Implementations in Lean to look at their Design Choices for Language Design. Special remarks are on the **higher order** capabilities, meaning Choreographies can take other Choreos as a parameter as some kind of sub-protocol, as well as **location polymorphism**. Location polymorphism here means the runtime substitution of locations.

1.4 Modular Compilation for Higher-Order Functional Choreographies

Higher order Choreographies make it non trivial to check which locations are involved in the Choreography. For example if Choreo C depends on Choreo K, the participants of C are only known after K is instantiated. This poses problems for static checks and the EPP. The paper proposes approaches to extend the lambda calcus to achieve a modular applications of an EPP (did not fully understand alot)

1.5 On the Monitorability of Session Types, in Theory and Practice

1.6 Multiparty Languages: The Choreographic and Multitier Cases

this work presents the two programming paradigms Choreographic programming and Multitier. Multitier being programm descriptions that specify the location of operations instead of explicit communication. While both approaches have different roots and went through different development, mainly because both workgroups rarely mix or push collective results, there still is a fundamental linkage between both. This similiaritys could lead both camps to **cross-fertilisation** that benefits both. This statement is backed by the comparison of two stripped down versions of an Choreo PL and a Multitier one.

1.7 Certified Automatic Repair of Uncompilable Protocols

There are Choreos that are not (directly) projectable to local Programs / types. Branching adds the challenges for the session, since continuation of a location protocal might depend on some choice that leads to different executions of two or more sub Protocols. This issue can be fixed by informing other Party members of branching choices that a location might does, called **knowledge of choice**. The paper also describes more clever ways of propagating choices, since not all choices are relevant to all locations and choices can be transitively propageted for example. The Automated Process is called **Amendment** or **repair**.

2 Proposal

Parallel and distributed systems have been established for decades by now and the relevance of communication in the field of programming has steadily increased ever since. While such Software Systems could be written with traditional PL's and paradigms there arises a whole ne set of programming error possibilities. For example deadlocks, or sending unexpected types of data to a receiver. To make programming DS more intuively and less error prone there has been high effort in creating specialized languages. those 'new' languages usally follow one of two paradigms. a Choreography or a Multiparty view. A Choreography tries to describes the communication from some kind of global view like a play or a dance with a construct of a directed send that specifies both sender and receiver. Multiier programming on the other hand lets the programmer shift the location of computation in code and thus masking the required communication. Still it has been shown that both approaches share strong similiaritys (Multiparty Languages: The Choreographic and Multitier Cases). Session Types, first introduced 1998 (i guess earlier?) and standardized by W3C in 2002, are an handy tool that has been used to statically check multiparty programs. They describe the communication layout of two or more partys and can already make strong safety guarantees by cheap static analysis. There has been many new research regarding session Types since (2015 i guess nochmal nachsehen) that focused on extending session types and their expressiveness,

but only a few cover session types in asychronous and unreliable environments. Fault tolrence in under these circumstances can be important, so i will investigate on the special challenges that arise here. Accompanying i implement a simple Choreography Model with session type projection in Lean4 that demonstrates some of my points

2.1 Research Questions

- unreliable nodes might be interesing? fault-tolerant and asynchronous systems were not covered by HasChor paper “and other functional Choreographic languages” do nothing to adress these difficulties (also mentioned in Modular Compilation for Higher-Order Functional Choreographies).
- how can session times still yield safety guarantees in asynchronous and/or unreliable systems (Multiparty Asynchronous Session Types Honda.)
- make Choreographys without obvious EPP projectable (like in Now it Compiles!). Are there other interesting problems than knowledge of choice?
- might be interesting to look into runtime verification with session types. (havent red alot about this, more in the future work / outlook sections)