Composable Inference Metaprogramming using Subproblems

by

Shivam Handa

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Signature redacted

Author ....
Department of Electrical Engineering and Computer Science
May 23, 2019

Signature redacted

Certified by....... ....
Martin Rinard
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Signature redacted

Accepted by........ ' UU ....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Composable Inference Metaprogramming using Subproblems

by

Shivam Handa

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2019, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

Inference metaprogramming enables effective probabilistic programming by supporting the decomposition of executions of probabilistic programs into subproblems and the deployment of hybrid probabilistic inference algorithms that apply different base probabilistic inference algorithms to different subproblems. I present the first sound and complete technique for extracting and stitching otherwise entangled subproblems for independent inference. I also prove asymptotic convergence results for hybrid inference algorithms for subproblem inference in probabilistic programs.

Thesis Supervisor: Martin Rinard
Title: Professor of Electrical Engineering and Computer Science

Contents

List of Figures

Chapter 1

Introduction

Probabilistic modeling and inference are now mainstream approaches deployed in many areas of computing and data analysis [33, 36, 22, 28, 12, 9]. To better support these computations, researchers have developed probabilistic programming languages, which include constructs that directly support probabilistic modeling and inference within the language itself [26, 14, 15, 23, 16, 40, 17, 38, 5, 20]. Probabilistic inference strategies provide the probabilistic reasoning required to implement these constructs.

It is well known that no one probabilistic inference strategy is appropriate for all probabilistic inference and modeling tasks [24]. Indeed, effective inference often involves breaking an inference problem down into subproblems, then applying different inference strategies to different subproblems as appropriate [24]. Applying this approach to probabilistic programs, specifically by specifying subtask decompositions and inference strategies to apply to each subtask, is called *inference metaprogramming.* Inference metaprogramming has been shown to dramatically improve the execution time and accuracy of probabilistic programs (in comparison with monolithic inference strategies that apply a single inference strategy to the entire program) [24].

1.1 Subproblems and Traces

When a probabilistic program executes, it produces a sample in the form of a *program trace.* Probabilistic inference algorithms for probabilisitic programs operate by changing stochastic choices in these traces to produce new traces. In this context, subproblems are subtraces and subproblem inference algorithms operate on these subtraces. The current state of the art defines subproblem inference as operating over the full program trace even though the inference algorithm should change only the subproblem [24]. This definition entangles the subproblem with the full program trace and complicates the implementation of the inference algorithms.

Independent Subproblems: I present a new technique that extracts each subproblem from the original program trace into its own independent trace. Inference is then performed over the full extracted trace, with the newly generated trace then stitched injected back into the original trace to complete the subproblem inference. By detangling the subproblem from the full trace, this approach simplifies the implementation of the inference algorithm and enforces the isolation of the inference algorithm within the target subproblem. It also enables the recursive application of inference metaprogramming to the extracted subtraces.

Successfully detangling the subproblem requires extracting a legal subtrace that an inference algorithm can successfully process. The first challenge is that the subtrace must be a valid trace of some probabilistic program, i.e., the subtrace must include all dependences required for the computation to be well defined and all deterministic computations must be correct within the trace. The second challenge is that the subtrace must not contain any stochastic choice outside the subproblem. To overcome these challenges, I present a new technique that appropriately converts outside stochastic choices into observe statements , then appropriately updates the extracted subtrace to reflect these changes. The dual stitching operation, which must reincorporate the newly inferred subtrace back into the original trace, then reverses the extraction while preserving the changes from the inference algorithm.

I present the technique in the context of a core probabilistic programming language based on the lambda calculus. I define an extraction operation that, given a subproblem defined over a current execution trace, extracts a corresponding subtrace. I also define a stitching algorithm that, given an inference result from the execution of the extracted subprogram, updates the execution trace to reflect the subproblem inference.

Soundness and Completeness: I present new soundness and completeness results for the extraction and stitching operations. The soundness result states that if the sequence subproblem extraction, inference over the extract subproblem trace, then stitching produces a new trace t, the direct subproblem inference applied to the subproblem entangled with the original trace can also produce the new trace t. The completeness result states that if direct subproblem inference applied to the subproblem entangled with the original trace can produce a new trace t, the the sequence subproblem extraction, inference over the extract subproblem trace, then stitching can also produce the new trace t.

1.2 Asymptotic Convergence

Probabilistic programs produce traces as samples from a distribution. Many probabilistic inference algorithms (such as Metropolis-Hastings [6] and Gibbs sampling [25]) take a sample as input and produce a new sample as output, with the new sample serving as input to the next iteration of the algorithm. A standard correctness property of such algorithms is *asymptotic convergence* — a guarantee that, in the limit as the number of iterations increases, the resulting sample will be drawn from the defined posterior distribution. Markov-Chain Monte-Carlo (MCMC) algorithms (which include both Metropolis-Hastings and Gibbs sampling) comprise a widely-used class of probabilistic inference algorithms that often come with asymptotic convergence guarantees.

Using inference metaprogramming to decompose and solve inference problems into subprograms produces new hybrid probabilistic inference algorithms. Whether or not these new hybrid inference algorithms (as implemented in the inference metaprogramming language) also asymptotically converge is often a question of interest (because it directly relates to the compositional soundness of the inference metaprogram).

I present a new asymptotic convergence result for inference metaprograms that apply asymptotically converging MCMC algorithms to appropriately defined subproblems. This result identifies a key restriction on the subproblem selection strategies that the inference metaprogram uses to identify subproblems. This restriction guarantees asymptotic convergence for inference metaprograms that apply a large class of asymptotically converging MCMC algorithms to the specified subproblems. This restriction requires:

- **Reversibility:** The subproblem selection strategy must be *reversible*, i.e., given a trace t that can be transformed into a new trace t' by applying the subproblem selection strategy to t, then applying the specified inference algorithm to the resulting subprogram to obtain the new trace t', it must also be possible to apply the subproblem selection strategy to t', then apply the inference algorithm to obtain the original trace t.

- **Connectivity:** The combination of all of the subproblem selection strategies in the inference metaprogram must connect the entire sample space, i.e., given any trace t, it must be possible to reach any other trace t' in the sample space by repeatedly applying subproblem selection selection strategies and specified inference algorithms.

1.3 Contributions

I claim the following contributions:

- **Independent Subproblems:** I present the first formulation of subproblem extraction for probabilistic programs. This formulation involves dual subproblem extraction and stitching operations. I state the first soundness and completeness properties that the combination of the extraction and stitching operations must satisfy and prove that my formulation satisfies these properties (Theorems 1 and 2).

- **Asymptotic Convergence:** I present the first asymptotic convergence result for hybrid probabilistic inference algorithms applied to subproblems in probabilistic programming languages (Theorem 11). This result characterizes subproblem selection strategies that guarantee asymptotic convergence for inference metaprograms that apply asymptotically converging MCMC algorithms to suproblems.

Effective probabilistic programming requires subproblem identification and hybrid probabilistic inference algorithms applied to the identified subproblems. The results in this paper enable the sound and complete decomposition of otherwise intractable probabilistic inference problems into tractable hybrid inference algorithms applied to subprograms. It also characterizes properties that entail asymptotic convergence of these resulting hybrid probabilistic inference algorithms.

Chapter 2

Language and Execution Model

I work with a core probabilistic programming language (Figure 2-1) based on the lambda calculus. A program in this language is a sequence of assume and observe statements. Expressions are derived from the untyped lambda calculus augmented with the $\mathsf{Dist}(e)$ expression, which allows the program to sample from a distribution Dist given parameter e.

$$\begin{array}{rcl} e_v \in E_v & := & x \mid \lambda.x\ e_v \mid (e_v\ e_v') \\ e, e_1, e_2 \in E & := & x \mid \lambda.x\ e \mid \mathsf{Dist}(e) \mid (e_1\ e_2) \\ s \in S & := & \mathsf{assume}\ x = e \mid \mathsf{observe}(\mathsf{Dist}(e) = e_v) \\ p \in P & := & \emptyset \mid s; p \end{array}$$

Figure 2-1: Probabilistic Lambda Calculus

$\mathsf{Dist}(e)$ can be seen as a set of probabilistic lambda calculus expressions $\{e_d | e_d \in \mathsf{Dist}(e) \subseteq E_v\}$. Based on the parameter expression e, $\mathsf{Dist}(e)$ makes a stochastic choice and returns an expression $e_v \in \mathsf{Dist}(e)$. I define:

$$\mathsf{Dist}(e)[x/y] = \mathsf{Dist}'(e[x/y]) = \{e_d[x/y] | e_d \in \mathsf{Dist}(e[x/y])\}$$

$$\mathsf{FreeVariables}(\mathsf{Dist}(e)) = \bigcup_{e_d \in \mathsf{Dist}(e) \cup \{e\}} \mathsf{FreeVariables}(e_d)$$

Because of the nondeterminism associated with stochastic choices, the execution strategy matters for the semantics of the language. I use call by value as the execution

$$\frac{}{x \rightarrow x} \qquad \frac{}{\lambda.x\ e \rightarrow \lambda.x\ e}$$

$$\frac{\begin{array}{c} e \rightarrow e' \\ e_v \in \mathsf{Dist}(e') \\ e_v \rightarrow e'_v \end{array}}{\mathsf{Dist}(e) \rightarrow e'_v} \qquad \frac{\begin{array}{c} e_1 \rightarrow e'_1 \\ e_2 \rightarrow e'_2 \\ e_1 \neq \lambda.x\ e \end{array}}{(e_1\ e_2) \rightarrow (e'_1\ e'_2)} \qquad \frac{\begin{array}{c} e_1 \rightarrow \lambda.x\ e \\ e_2 \rightarrow e'_2 \\ e[e'_2/x] \rightarrow e' \end{array}}{(e_1\ e_2) \rightarrow e'}$$

$$\frac{}{\emptyset \rightarrow \emptyset} \qquad \frac{\begin{array}{c} e \rightarrow e' \\ p[e'/x] \rightarrow p' \end{array}}{\mathsf{assume}\ x = e; p \rightarrow p'} \qquad \frac{e \rightarrow e' \quad p \rightarrow p'}{\begin{array}{c} \mathsf{observe}(\mathsf{Dist}(e) = e_v); p \rightarrow \\ \mathsf{observe}(\mathsf{Dist}(e') = e_v); p' \end{array}}$$

Figure 2-2: Execution Strategy for Probabilistic Programs

strategy and forbid the reduction of expressions within a lambda. Figure 2-2 presents the execution strategy.

Traces: When my framework execute a program, it produce a trace of its execution (Figure 2-3). This trace records the executed sequence of assume and observe commands, including the value of each evaluated (sub)expression. It also assigns a unique identifier to each evaluated (sub)expression and stochastic choice. These identifiers will be later used to construct a dependence graph which helps in defining valid subproblems.

$$\begin{array}{rcl}
v \in V & := & x \mid \langle \lambda.x\ e, \sigma_v, \sigma_{id} \rangle \mid (v_1\ v_2) \\
aa \in aA & := & \perp\ \mid x = ae \\
ae \in aE & := & (x : x)\#id \mid (x(id') : v)\#id \\
 & \mid & (\lambda.x\ e : v)\#id \mid ((ae_1\ ae_2)aa : v)\#id \\
 & \mid & (\mathsf{Dist}(ae\#id') = ae' : v)\#id \\
as \in aS & := & \mathsf{assume}\ x = ae \mid \mathsf{observe}(\mathsf{Dist}(ae\#id) = e_v) \\
t \in T & := & \emptyset \mid as; t
\end{array}$$

Figure 2-3: Traces

Two traces are equal if and only if they differ in the choice of unique identifiers selected for each augmented expression and stochastic choice.

I define the execution, including the generation of valid traces t, with the transition relation $\Rightarrow_s \subseteq \Sigma_v \times \Sigma_{id} \times P \rightarrow T$ (Figure 2-4). Conceptually, the transition relation executes program p (in accordance to my execution strategy defined in Figure 2-2) under the environment σ_v, σ_{id} to obtain a trace t, where $\sigma_v : Vars \rightarrow V$ and $\sigma_{id} : Vars \rightarrow ID$. σ_v is a map from variable name to its corresponding assigned value, whereas σ_{id} gives the id of the expression which assigned this value to that variable.

Given a program p, I define the set of all valid traces which can be obtained by executing p as $T_p = \mathsf{Traces}(p)$.

$$t \in \mathsf{Traces}(p) \iff \emptyset, \emptyset \vdash p \Rightarrow_s t$$

A trace contains all the information about the underlying program from which the trace was generated. Given a trace, we can drop the computed values and assigned ids and reroll the augmented expressions to recover the underlying program. The transition relation $\Rightarrow_r \subseteq T \rightarrow P$ (Figure 2-5) formalizes this procedure. Given a trace t, I define

$$p = \mathsf{Program}(t) \iff t \Rightarrow_r p$$

Note that $\forall\ t, p.\ t \in \mathsf{Traces}(p) \implies p = \mathsf{Program}(t)$. The reverse may not be true as there are additional constraints that valid traces must satisfy.

Dependence Graphs: Given a trace t, I define the dependence graph $\langle \mathcal{N}, \mathcal{D}, \mathcal{E} \rangle = \mathsf{Graph}(t)$ as a 3-tuple $\langle \mathcal{N}, \mathcal{D}, \mathcal{E} \rangle$ where $\mathcal{N} : \mathsf{ID} \rightarrow \{\perp, \mathsf{Sample}\}$ is a map from ID to either $\perp$ (when the corresponding augmented expression for an $id \in ID$ is a deterministic computation) or Sample (when the augmented expression for an $id \in ID$ makes a stochastic choice).

$\mathcal{D} \subseteq \mathsf{ID} \times \mathsf{ID}$ are data dependence edges. There is a data dependence edge $\langle id_1, id_2 \rangle \in \mathcal{D}$ if the value of the augmented expression id_2 directly depends on the augmented expression id_1.

$\mathcal{E} \subseteq \mathsf{ID} \times \mathsf{ID}$ are existential edges. There is a existential edge $\langle id_1, id_2 \rangle \in \mathcal{E}$ if the value of the augmented expression id_1 controls whether or not an augmented expression id_2 executed. For example, in a lambda application $(ae_1\ ae_2)x = ae_3$, all augmented expressions in ae_3 were executed only because of the value of ae_1.

$$\frac{id \leftarrow \textsf{Fresh ID} \qquad y \notin \textsf{dom } \sigma_v}{\sigma_v, \sigma_{id} \vdash y \Rightarrow_s y, id, (y : y)\#id}$$

$$\frac{id \leftarrow \textsf{Fresh ID} \qquad x \in \textsf{dom } \sigma_v}{\sigma_v, \sigma_{id} \vdash x \Rightarrow_s \sigma_v(x), id, (x(\sigma_{id}(x)) : \sigma_v(x))\#id}$$

$$\frac{\begin{array}{c} id \leftarrow \textsf{Fresh ID} \\ \sigma'_v = \textsf{RestrictKeys}(\sigma_v, \textsf{FreeVariables}(\lambda.x\ e)) \\ \sigma'_{id} = \textsf{RestrictKeys}(\sigma_{id}, \textsf{FreeVariables}(\lambda.x\ e)) \\ v = \langle \lambda.x\ e, \sigma'_v, \sigma'_{id} \rangle \end{array}}{\sigma_v, \sigma_{id} \vdash \lambda.x\ e \Rightarrow_s v, id, (\lambda.x\ e : v)\#id}$$

$$\frac{\begin{array}{c} id \leftarrow \textsf{Fresh ID} \qquad id' \leftarrow \textsf{Fresh ID} \\ \sigma_v, \sigma_{id} \vdash e \Rightarrow_s v, id_e, ae \\ e'_v \in \textsf{Dist}(v) \qquad \sigma_v, \sigma_{id} \vdash e'_v \Rightarrow_s v, id_v, ae_v \end{array}}{\sigma_v, \sigma_{id} \vdash \textsf{Dist}(e) \Rightarrow_s v, id, (\textsf{Dist}(ae\#id') = ae_v : v)\#id}$$

$$\frac{\begin{array}{c} id \leftarrow \textsf{Fresh ID} \qquad x \leftarrow \textsf{Fresh variable name} \\ \sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \langle \lambda.y\ e, \sigma'_v, \sigma'_{id} \rangle, id_1, ae_1 \\ \sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s v', id_2, ae_2 \\ \sigma'_v[x \to v'], \sigma'_{id}[x \to id_2] \vdash e[x/y] \Rightarrow_s v, id_e, ae_e \end{array}}{\sigma_v, \sigma_{id} \vdash (e_1\ e_2) \Rightarrow_s v, id, ((ae_1\ ae_2)x = ae_e : v)\#id}$$

$$\frac{\begin{array}{c} id \leftarrow \textsf{Fresh ID} \\ \sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s v_1, id_1, ae_1 \qquad \sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s v_2, id_2, ae_2 \\ v_1 \neq \langle \lambda.x\ e, \sigma'_v, \sigma'_{id} \rangle \qquad v = (v_1\ v_2) \end{array}}{\sigma_v, \sigma_{id} \vdash (e_1\ e_2) \Rightarrow_s v, id, ((ae_1\ ae_2) \perp : v)\#id}$$

(a) Executing expressions, $\Rightarrow_s \subseteq \Sigma_v \times \Sigma_{id} \times E \to V \times ID \times aE$

$$\frac{}{\sigma_v, \sigma_{id} \vdash \emptyset \Rightarrow_s \emptyset}$$

$$\frac{\sigma_v, \sigma_{id} \vdash e \Rightarrow_s v, id, ae \qquad \sigma_v[x \to v], \sigma_{id}[x \to id] \vdash p \Rightarrow_s t}{\sigma_v, \sigma_{id} \vdash \textsf{assume } y = e; p \Rightarrow_s \textsf{assume } x = ae; t}$$

$$\frac{\begin{array}{c} id \leftarrow \textsf{Fresh ID} \\ \sigma_v, \sigma_{id} \vdash e \Rightarrow_s e'_v, id_e, ae \qquad \sigma_v, \sigma_{id} \vdash p \Rightarrow_s t \end{array}}{\sigma_v, \sigma_{id} \vdash \textsf{observe}(\textsf{Dist}(e) = e_v); p \Rightarrow_s \textsf{observe}(\textsf{Dist}(ae\#id) = e_v); t}$$

(b) Executing Programs, $\Rightarrow_s \subseteq \Sigma_v \times \Sigma_{id} \times P \to T$

Figure 2-4: Valid Traces

$$\frac{}{(x : x)\#id \Rrightarrow_r x} \qquad \frac{}{(x(id') : v)\#id \Rrightarrow_r x}$$

$$\frac{}{(\lambda.x\ e : v)\#id \Rrightarrow_r \lambda.x\ e} \qquad \frac{ae_1 \Rrightarrow_r e_1 \qquad ae_2 \Rrightarrow_r e_2}{((ae_1\ ae_2)aa : v)\#id \Rrightarrow_r (e_1\ e_2)}$$

$$\frac{ae \Rrightarrow_r e}{(\mathsf{Dist}(ae\#id') = ae' : v)\#id \Rrightarrow_r \mathsf{Dist}(e)}$$

(a) Rolling back Augmented Expressions, $\Rrightarrow_r \subseteq aE \to E$

$$\frac{}{\emptyset \Rrightarrow_r \emptyset} \qquad \frac{ae \Rrightarrow_r e \qquad t \Rrightarrow_r p}{\mathsf{assume}\ x = ae; t \Rrightarrow_r \mathsf{assume}\ x = e; p}$$

$$\frac{ae \Rrightarrow_r e \qquad t \Rrightarrow_r p}{\mathsf{observe}(\mathsf{Dist}(ae\#id) = e_v); t \Rrightarrow_r \mathsf{observe}(\mathsf{Dist}(e) = e_v); p}$$

(b) Rolling back Traces, $\Rrightarrow_r \subseteq T \to P$

Figure 2-5: Rolling back Traces to Probabilistic Program

Changing the value of ae_1 would require dropping the augmented expression ae_3 and recomputing another expression based on the new value of ae_1.

I formalize the dependence graph generation procedure as a transition relation $\Rrightarrow_g \subseteq T \to \langle \mathcal{N}, \mathcal{D}, \mathcal{E} \rangle$ (Figure 2-6) . I use the shorthand $\langle \mathcal{N}, \mathcal{D}, \mathcal{E} \rangle = \mathsf{Graph}(t)$ if $\langle \mathcal{N}, \mathcal{D}, \mathcal{E} \rangle$ is the dependence graph for trace t i.e. $t \Rrightarrow_g \langle \mathcal{N}, \mathcal{D}, \mathcal{E} \rangle$.

Valid Subproblems: Subproblem inference must 1) change only the identified subproblem and not the enclosing trace while 2) producing a valid trace for the full probabilistic program. Valid subproblems must therefore include all parts of the trace that may change if any part of the subproblem changes. I formalize this requirement as follows.

Given a trace t with dependence graph $\langle \mathcal{N}, \mathcal{D}, \mathcal{E} \rangle = \mathsf{Graph}(t)$, a valid subproblem $\mathcal{S} \subseteq \mathsf{dom}\ \mathcal{N}$ must satisfy two properties: 1) there are no outgoing existential edges and 2) all outgoing data dependence edges must terminate at a stochastic choice (Sample node).

The first property ensures that parts of the trace which were executed due to values of expressions in the subproblem are also part of the subproblem. An example of this is lambda evaluation $(ae_1\ ae_2)x = ae_3$. If the value of ae_1 can be changed by

$$\frac{}{(x : x)\#id \Rightarrow_g id, \langle \{id \to \bot\}, \emptyset, \emptyset \rangle} \qquad \frac{}{(x(id') : v)\#id \Rightarrow_g id, \langle \{id \to \bot\}, \{\langle id', id\rangle\}, \emptyset \rangle}$$

$$\frac{}{\begin{array}{c}(\lambda.x\ e : \langle \lambda.x\ e, \sigma_v, \sigma_{id}\rangle)\#id \Rightarrow_g id, \\ \langle \{id \to \bot\}, \emptyset, \emptyset \rangle\end{array}}$$

$$\frac{\begin{array}{c}ae_1 \Rightarrow_g id_1, \langle \mathcal{N}_1, \mathcal{D}_1, \mathcal{E}_1\rangle \qquad ae_2 \Rightarrow_g id_2, \langle \mathcal{N}_2, \mathcal{D}_2, \mathcal{E}_2\rangle \qquad ae \Rightarrow_g id_e, \langle \mathcal{N}_e, \mathcal{D}_e, \mathcal{E}_e\rangle \\ \langle \mathcal{N}, \mathcal{D}, \mathcal{E}\rangle = \langle \mathcal{N}_1 \cup \mathcal{N}_2 \cup \mathcal{N}_e, \mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_e, \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_e\rangle\end{array}}{\begin{array}{c}((ae_1\ ae_2)x = ae : v)\#id \Rightarrow_g \\ id, \langle \mathcal{N}[id \to \bot], \mathcal{D} \cup \{\langle id_1, id\rangle, \langle id_e, id\rangle\}, \mathcal{E} \cup \{\langle id_1, id_n\rangle | id_n \in \mathsf{dom}\ \mathcal{N}_e\}\rangle\end{array}}$$

$$\frac{ae_1 \Rightarrow_g id_1, \langle \mathcal{N}, \mathcal{D}, \mathcal{E}\rangle \qquad ae_2 \Rightarrow_g id_2, \langle \mathcal{N}', \mathcal{D}', \mathcal{E}'\rangle}{\begin{array}{c}((ae_1\ ae_2)\ \bot : v)\#id \Rightarrow_g \\ id, \langle \mathcal{N} \cup \mathcal{N}'[id \to \bot], \mathcal{D} \cup \mathcal{D}'\{\langle id_1, id\rangle, \langle id_2, id\rangle\}, \mathcal{E} \cup \mathcal{E}\rangle\end{array}}$$

$$\frac{\begin{array}{c}ae \Rightarrow_g id_e, \langle \mathcal{N}, \mathcal{D}, \mathcal{E}\rangle \qquad ae' \Rightarrow_g id'_e, \langle \mathcal{N}', \mathcal{D}', \mathcal{E}'\rangle \\ \langle \mathcal{N}_r, \mathcal{D}_r, \mathcal{E}_r\rangle = \\ \langle \mathcal{N} \cup \mathcal{N}'[id' \to \mathsf{Sample}], \mathcal{D} \cup \mathcal{D}' \cup \{\langle id_e, id'\rangle\}, \mathcal{E} \cup \mathcal{E}' \cup \{\langle id', id_n\rangle | id_n \in \mathsf{dom}\ \mathcal{N}'\}\rangle\end{array}}{\begin{array}{c}(\mathsf{Dist}(ae\#id') = ae' : v)\#id \Rightarrow_g \\ id, \langle \mathcal{N}_r[id \to \bot], \mathcal{D}_r \cup \{\langle id'_e, id\rangle, \langle id', id\rangle\}, \mathcal{E}_r\rangle\end{array}}$$

(a) Dependence Graph generation for augmented Expressions $\Rightarrow_g \subseteq aE \to ID \times \langle \mathcal{N}, \mathcal{D}, \mathcal{E}\rangle$

$$\frac{ae \Rightarrow_g id, \langle \mathcal{N}, \mathcal{D}, \mathcal{E}\rangle}{\mathsf{assume}\ x = ae \Rightarrow_g \langle \mathcal{N}, \mathcal{D}, \mathcal{E}\rangle}$$

$$\frac{ae \Rightarrow id', \langle \mathcal{N}, \mathcal{D}, \mathcal{E}\rangle}{\mathsf{observe}(\mathsf{Dist}(ae\#id) = e_v) \Rightarrow_g \langle \mathcal{N}[id \to \mathsf{Sample}], \mathcal{D} \cup \{\langle id', id\rangle\}, \mathcal{E}\rangle}$$

(b) Dependence Graph generation for augmented Statements, $\Rightarrow_g \subseteq aS \to \langle \mathcal{N}, \mathcal{D}, \mathcal{E}\rangle$

$$\frac{}{\emptyset \Rightarrow_g \langle \emptyset, \emptyset, \emptyset\rangle} \qquad \frac{as \Rightarrow_g \langle \mathcal{N}_s, \mathcal{D}_s, \mathcal{E}_s\rangle \qquad t \Rightarrow_g \langle \mathcal{N}, \mathcal{D}, \mathcal{E}\rangle}{as; t \Rightarrow_g \langle \mathcal{N} \cup \mathcal{N}_s, \mathcal{D} \cup \mathcal{D}_s, \mathcal{E} \cup \mathcal{E}_s\rangle}$$

(c) Dependence Graph generation for Traces, $\Rightarrow_g \subseteq T \to \langle \mathcal{N}, \mathcal{D}, \mathcal{E}\rangle$

Figure 2-6: Dependency Graph for a Trace t

the subproblem inference, ae_3 may or may not exist. Hence ae_3 should be within the subproblem to ensure that the inference algorithm can change it if necessary.

The second property ensures that any change made by the subproblem inference can be absorbed by a stochastic choice. For example, when the internal parameter of a Dist changes, one can absorb the change by changing the probability of the trace to account for the change in the probability of the value generated by the execution of the absorbing Dist node. The changes are absorbed by the stochastic choice and do not propagate further into the remaining parts of the trace outside the subproblem.

I formalize the two properties as follows:

- $\forall id \in \mathcal{S}.\ \langle id, id_o \rangle \in \mathcal{E} \implies id_o \in \mathcal{S}$
- $\forall id \in \mathcal{S}.\ \langle id, id_o \rangle \in \mathcal{D} \wedge id_o \in \mathsf{dom}\ \mathcal{N} - \mathcal{S} \implies \mathcal{N}(id_o) = \mathsf{Sample}$

The absorbing set $\mathcal{A} \subseteq \mathsf{dom}\ \mathcal{N} - \mathcal{S}$ of a subproblem $\mathcal{S}$ is the set of stochastic choices whose value directly depends on the nodes in the subproblem i.e. $\mathcal{A} = \{id_a | id_a \in \mathsf{dom}\ \mathcal{N} - \mathcal{S} \wedge \exists\ id_i \in \mathcal{S}.\ \langle id_i, id_o \rangle \in \mathcal{D}\}$.

The input boundary $\mathcal{B} \subseteq \mathsf{dom}\ \mathcal{N} - \mathcal{S}$ of a subproblem $\mathcal{S}$ is the set of nodes on which the subproblem directly depends on i.e. $\mathcal{B} = \{id_b | id_b \in \mathsf{dom}\ \mathcal{N} - \mathcal{S} \wedge \forall\ id_i \in \mathcal{S}.\ \langle id_b, id_i \rangle \in \mathcal{D}\}$.

Entangled Subproblem Inference: Following [24], I define *entangled subproblem inference* using the infer procedure [24], which takes as parameters a subproblem selection strategy SS, an inference tactic IT, and an input trace t. The subproblem inference mutates t to produce a new trace t'.

$$\frac{\mathsf{SS}(t) = \mathcal{S} \qquad t' = \mathsf{IT}(t, \mathcal{S}) \\ t' \in \mathsf{Traces}(\mathsf{Program}(t)) \qquad \mathcal{S} \vdash t \equiv t'}{\mathsf{infer}(\mathsf{SS}, \mathsf{IT}, t) \Rightarrow_i t'}$$

This formulation works with arbitrary subproblem selection strategies SS. The requirement is that, given a trace t, SS must produce a valid subproblem $\mathcal{S}$ over t. I also work with inference algorithms IT that take as input a full program trace t and a valid subproblem $\mathcal{S}$ and return a mutated full program trace t'. I require that the output trace t' 1) is from the same program as the trace t and 2) t' differs from t only in a) the stochastic choices from the subproblem $\mathcal{S}$ and b) the deterministic computations that depend on these stochastic choices. I formalize these constraints as

- $t' \in \mathsf{Traces}(\mathsf{Program}(t))$
- $\mathcal{S} \vdash t \equiv t'$

Figure 2-7 presents the definition of $\equiv$. Note that operating with entangled subproblems forces the inference tactic IT to take the full program trace t as a parameter even though it must modify at most only the subproblem.

$$\frac{}{\mathcal{S} \vdash (x : x)\#id \equiv (x : x)\#id'} \qquad \frac{}{\mathcal{S} \vdash (x(id_v) : v)\#id \equiv (x(id'_v) : v')\#id'}$$

$$\frac{}{\mathcal{S} \vdash (\lambda.x\ e : v)\#id \equiv (\lambda.x\ e : v')\#id'}$$

$$\frac{\mathsf{ID}(ae_1) \notin \mathcal{S} \quad \mathcal{S} \vdash ae_1 \equiv ae'_1 \quad \mathcal{S} \vdash ae'_2 \equiv ae'_2 \quad \mathcal{S} \vdash ae'_3 \equiv ae'_3}{\mathcal{S} \vdash ((ae_1\ ae_2)x = ae_3 : v)\#id \equiv ((ae'_1\ ae'_2)x = ae'_3 : v')\#id}$$

$$\frac{\mathsf{ID}(ae_1) \notin \mathcal{S} \quad \mathcal{S} \vdash ae_1 \equiv ae'_1 \quad \mathcal{S} \vdash ae'_2 \equiv ae'_2}{\mathcal{S} \vdash ((ae_1\ ae_2) \perp: v)\#id \equiv ((ae'_1\ ae'_2) \perp: v')\#id'}$$

$$\frac{\mathsf{ID}(ae_1) \in \mathcal{S} \quad \mathcal{S} \vdash ae_1 \equiv ae'_1 \quad \mathcal{S} \vdash ae'_2 \equiv ae'_2}{\mathcal{S} \vdash ((ae_1\ ae_2)aa : v)\#id \equiv ((ae'_1\ ae'_2)aa' : v')\#id'}$$

$$\frac{id_e \notin \mathcal{S} \quad \mathcal{S} \vdash ae \equiv ae' \quad \mathcal{S} \vdash ae_v \equiv ae'_v}{\mathcal{S} \vdash (\mathsf{Dist}(ae\#id_e) = ae_e : v)\#id \equiv (\mathsf{Dist}(ae'\#id_e) = ae'_e : v')\#id'}$$

$$\frac{id_e \in \mathcal{S} \quad \mathcal{S} \vdash ae \equiv ae'}{\mathcal{S} \vdash (\mathsf{Dist}(ae\#id_e) = ae_e : v)\#id \equiv (\mathsf{Dist}(ae'\#id'_e) = ae'_e : v')\#id'}$$

(a) Equivalence Check over augmented expressions, $\equiv \subseteq \mathcal{P}(ID) \times \mathcal{P}(ID) \times aE \times aE$

$$\frac{}{\mathcal{S} \vdash \emptyset \equiv \emptyset} \qquad \frac{\mathcal{S} \vdash ae \equiv ae' \quad \mathcal{S} \vdash t \equiv t'}{\mathcal{S} \vdash \mathsf{assume}\ x = ae; t \equiv \mathsf{assume}\ x = ae'; t'}$$

$$\frac{\mathcal{S} \vdash ae \equiv ae' \quad \mathcal{S} \vdash t \equiv t'}{\mathcal{S} \vdash \mathsf{observe}(\mathsf{Dist}(ae\#id) = e_v); t \equiv \mathsf{observe}(\mathsf{Dist}(ae'\#id) = e_v); t'}$$

(b) Equivalence check over traces, $\equiv \subseteq \mathcal{P}(ID) \times \mathcal{P}(ID) \times T \times T$

Figure 2-7: Equivalence Check for correct Inference

Chapter 3

Independent Subproblem Inference

The basic idea of independent subproblem inference is to extract an independent subtrace t_s from the original trace t given a subproblem $\mathcal{S}$, perform inference over the extracted subtrace t_s to obtain a new trace t'_s, then stitch t'_s back into t. Here, consistent with standard inference techniques for probabilistic programs [24], t_s and t'_s are valid traces of the same program p_s (the subprogram for the subtraces t_s and t'_s). The key challenge is converting the entangled subproblem (which is typically incomplete and therefore not a valid trace of any program) into a valid trace by transforming the subproblem to include external dependences and correctly scope both internal and external dependences in the extracted trace without given the inference algorithm access to any external stochastic choices (including latent choices nested inside certain lambda expressions which would otherwise override choices outside the subproblem) which it must not change.

Extract Trace: I define the extraction procedure $t_s = \mathsf{ExtractTrace}(t, \mathcal{S})$ using the transition relation $\Rightarrow_{ex} \subseteq \mathcal{P}(ID) \times \mathcal{P}(ID) \times T \rightarrow T$ (Figure 3-1).

The extraction procedure removes $\mathsf{Dist}(ae\#id) = ae_v$ augmented expressions which are not within the subproblem and converts them into $\mathsf{observe}$ statements. This transformation constrains the value of these stochastic choices to the values present in the original trace. It leaves the stochastic choices in the subproblem in place and therefore accessible to the inference algorithm.

For augmented expressions of the form $(ae_1\ ae_2)y = ae_3$, when ae_1 is within the

$$\frac{}{\mathcal{S} \vdash (x(id') : v)\#id \Rightarrow_{ex} (x(id') : v)\#id, \emptyset}$$

$$\frac{}{\mathcal{S} \vdash (x : x)\#id \Rightarrow_{ex} (x : x)\#id, \emptyset}$$

$$\frac{}{\mathcal{S} \vdash (\lambda.x\ e : v)\#id \Rightarrow_{ex} (\lambda.x\ e : v)\#id, \emptyset}$$

$$\frac{\mathsf{ID}(ae_1) \in \mathcal{S} \qquad \mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae'_1, t_s \qquad \mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae'_2, t'_s}{\mathcal{S} \vdash ((ae_1\ ae_2)aa : v)\#id \Rightarrow_{ex} ((ae'_1\ ae'_2)aa : v)\#id, t_s; t'_s}$$

$$\frac{\mathsf{ID}(ae_1) \notin \mathcal{S} \qquad \mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae'_1, t_s \qquad \mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae'_2, t'_s}{\mathcal{S} \vdash ((ae_1\ ae_2) \perp : v)\#id, t_p \Rightarrow_{ex} ((ae'_1\ ae'_2) \perp : v)\#id, t_s; t'_s}$$

$$\frac{\begin{array}{c} \mathsf{ID}(ae_1) \notin \mathcal{S} \qquad \mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae'_1, t_s \\ \mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae'_2, t'_s \qquad \mathcal{S} \vdash ae_3 \Rightarrow_{ex} ae'_3, t''_s \\ x \leftarrow \text{Fresh variable name} \\ t'''_s = t_s; \text{assume } x = ae'_1; t'_s; \text{assume } y = ae'_2; t''_s \end{array}}{\mathcal{S} \vdash ((ae_1\ ae_2)y = ae_3 : v)\#id, t_p \Rightarrow_{ex} ae'_3, t'''_s}$$

$$\frac{id' \notin \mathcal{S} \qquad \mathcal{S} \vdash ae \Rightarrow_{ex} ae', t_s \qquad ae_v \Rightarrow_r e_v \qquad \mathcal{S} \vdash ae_v \Rightarrow_{ex} ae'_v, t'_s}{\mathcal{S} \vdash (\mathsf{Dist}(ae\#id') = ae_v : v)\#id \Rightarrow_{ex} ae'_v, t_s; \mathsf{observe}(\mathsf{Dist}(ae'\#id') = e_v); t'_s}$$

$$\frac{id' \in \mathcal{S} \qquad \mathcal{S} \vdash ae \Rightarrow_{ex} ae', t_s}{\mathcal{S} \vdash (\mathsf{Dist}(ae\#id') = ae_v : v)\#id \Rightarrow_{ex} (\mathsf{Dist}(ae'\#id') = ae_v : v)\#id, t_s}$$

(a) Extracting subtrace from an Augmented Expression, $\Rightarrow_{ex} \subseteq \mathcal{P}(ID) \times \mathcal{P}(ID) \times aE \rightarrow aE \times T$

$$\frac{}{\mathcal{S} \vdash \emptyset \Rightarrow_{ex} \emptyset} \qquad \frac{\begin{array}{c} id = \mathsf{ID}(ae) \\ \mathcal{S} \vdash ae \Rightarrow_{ex} ae', t_s \qquad \mathcal{S} \vdash t \Rightarrow_{ex} t'_s \end{array}}{\mathcal{S} \vdash \text{assume } x = ae; t \Rightarrow_{ex} t_s; \text{assume } x = ae'; t'_s}$$

$$\frac{\mathcal{S} \vdash ae \Rightarrow_{ex} ae', t_s \qquad \mathcal{S} \vdash t \Rightarrow_{ex} t'_s}{\mathcal{S} \vdash \mathsf{observe}(\mathsf{Dist}(ae\#id) = e_v); t \Rightarrow_{ex} t_s; \mathsf{observe}(\mathsf{Dist}(ae'\#id) = e_v); t'_s}$$

(b) Extracting subtrace from a given trace, $\Rightarrow_{ex} \subseteq \mathcal{P}(ID) \times \mathcal{P}(ID) \times T \rightarrow T$

Figure 3-1: Extraction Relation

$$\frac{}{\mathcal{S} \vdash (x : v)\#id, (x : v')\#id', t_p \Rightarrow_{st} (x : v')\#id', t_p}$$

$$\frac{}{\mathcal{S} \vdash (x(id_v) : v)\#id, (x(id'_v) : v')\#id', t_p \Rightarrow_{st} (x(id'_v) : v')\#id', t_p}$$

$$\frac{}{\mathcal{S} \vdash (\lambda.x\ e : v)\#id', (\lambda.x\ e : v')\#id, t_p \Rightarrow_{st} (\lambda.x\ e : v')\#id, t_p}$$

$$\frac{\mathsf{ID}(ae'_1) \in \mathcal{S} \qquad \mathcal{S} \vdash ae'_2, ae_2, t_p \Rightarrow_{st} ae''_2, t'_p \qquad \mathcal{S} \vdash ae'_1, ae_1, t'_p \Rightarrow_{st} ae''_1, t''_p}{\mathcal{S} \vdash ((ae'_1\ ae'_2)aa' : v')\#id', ((ae_1\ ae_2)aa : v)\#id, t_p \Rightarrow_{st} ((ae''_1\ ae''_2)aa : v)\#id, t''_p}$$

$$\frac{\begin{array}{c}\mathsf{ID}(ae'_1) \notin \mathcal{S} \qquad \mathcal{S} \vdash ae'_3, ae_3, t_p \Rightarrow_{st} ae''_3, t'_p; \mathsf{assume}\ y = ae_2 \\ \mathcal{S} \vdash ae'_2, ae_2, t'_p \Rightarrow_{st} ae''_2, t''_p; \mathsf{assume}\ x = ae_1 \qquad \mathcal{S} \vdash ae'_1, ae_1, t''_p \Rightarrow_{st} ae''_1, t'''_p\end{array}}{\mathcal{S} \vdash ((ae'_1\ ae'_2)y = ae'_3 : v')\#id, ae_3, t_p \Rightarrow_{st} ((ae''_1\ ae''_2)y = ae''_3 : \mathcal{V}(ae''_3))\#id, t'''_p}$$

$$\frac{\mathsf{ID}(ae'_1) \notin \mathcal{S} \qquad \mathcal{S} \vdash ae'_2, ae_2, t_p \Rightarrow_{st} ae''_2, t'_p \qquad \mathcal{S} \vdash ae'_1, ae_1, t'_p \Rightarrow_{st} ae''_1, t''_p}{\mathcal{S} \vdash ((ae'_1\ ae'_2) \perp : v')\#id', ((ae_1\ ae_2) \perp : v)\#id, t_p \Rightarrow_{st} ((ae''_1\ ae''_2) \perp : v)\#id, t''_p}$$

$$\frac{id'_v \in \mathcal{S} \qquad \mathcal{S} \vdash ae', ae, t_p \Rightarrow_{st} ae'', t'_p}{\begin{array}{c}\mathcal{S} \vdash (\mathsf{Dist}(ae'\#id'_v) = ae'_v : v')\#id', (\mathsf{Dist}(ae\#id_v) = ae_v : v)\#id, t_p \Rightarrow_{st} \\ (\mathsf{Dist}(ae\#id_v) = ae_v : v)\#id, t'_p\end{array}}$$

$$\frac{\begin{array}{c}id'_v \notin \mathcal{S} \qquad \mathcal{S} \vdash ae'_v, ae_v, t_p \Rightarrow_{st} ae''_v, t'_p; \mathsf{observe}(\mathsf{Dist}(ae\#id'_v) = e_v) \\ \mathcal{S} \vdash ae', ae, t'_p \Rightarrow_{st} ae'', t''_p\end{array}}{\mathcal{S} \vdash (\mathsf{Dist}(ae'\#id_v) = ae'_v : v')\#id', ae_v, t_p \Rightarrow_{st} (\mathsf{Dist}(ae\#id_v) = ae''_v : \mathcal{V}(ae''_v))\#id, t''_p}$$

(a) Stitching augmented expressions, $\Rightarrow_{st} \subseteq \mathcal{P}(ID) \times \mathcal{P}(ID) \times aE \times aE \times T \rightarrow aE \times T$

$$\frac{}{\mathcal{S} \vdash \emptyset, \emptyset \Rightarrow_{st} \emptyset} \qquad \frac{\mathcal{S} \vdash ae, ae', t_p \Rightarrow_{st} ae'', t'_p \qquad \mathcal{S} \vdash t, t'_p \Rightarrow_{st} t'_s}{\mathcal{S} \vdash t; \mathsf{assume}\ x = ae, t_p; \mathsf{assume}\ x = ae' \Rightarrow_{st} \mathsf{assume}\ x = ae''; t'_s}$$

$$\frac{\mathcal{S} \vdash ae, ae', t_p \Rightarrow_{st} ae'', t'_p \qquad \mathcal{S} \vdash t, t'_p \Rightarrow_{st} t'_s}{\begin{array}{c}\mathcal{S} \vdash t; \mathsf{observe}(\mathsf{Dist}(ae\#id) = e_v), t_p; \mathsf{observe}(\mathsf{Dist}(ae'\#id') = e_v) \Rightarrow_{st} \\ \mathsf{observe}(\mathsf{Dist}(ae''\#id) = e_v); t'_s\end{array}}$$

(b) Stitching trace and subtrace, $\Rightarrow_{st} \subseteq \mathcal{P}(ID) \times \mathcal{P}(ID) \times T \times T \rightarrow T$

Figure 3-2: Stitching Transition Relation

subproblem, its value can change and hence existential edges place the augmented expressions in ae_3 within the subproblem. When ae_1 is not within the subproblem, then some stochastic choices may or not be within the subproblem. If I keep the augmented expression as is, the inference algorithm may unroll ae_3 and execute it again, changing some stochastic choices in ae_3 but not in the subproblem. If I modify ae_3, the constraint of ae_3 being a valid lambda application breaks. I solve this problem by introducing assume statements and correctly scoping the resulting dependences.

Stitch Trace: Given a trace t, a valid subproblem $\mathcal{S}$ over the trace and a subtrace t_s, the stitching procedure stitches back the trace t_s into t to get a new trace t'. I define the stitching procedure $t' = \mathsf{StitchTrace}(t, t_s, \mathcal{S})$ using a transition relation $\Rightarrow_{st} \subseteq \mathcal{P}(ID) \times \mathcal{P}(ID) \times T \times T \rightarrow T$ (Figure 3-2). Stitching is the dual of extraction. It uses the original trace to figure out the structure of the resultant trace and stitches back the expressions to get a new trace t'.

Independent Inference: I define independent subproblem inference using the infer procedure. infer takes as input a subproblem selection strategy SS, a trace t and an inference tactic IT. This differs from tangled inference in that inference tactic IT takes only the extracted subtrace as input and not the entire program trace. This approach enables the use of standard inference algorithms which are designed to operate on complete traces (and not entangled subproblems).

The new inference procedure works as described below:

$$\frac{\mathsf{SS}(t) = \mathcal{S} \qquad t_s = \mathsf{ExtractTrace}(t, \mathcal{S}) \qquad t'_s = \mathsf{IT}(t_s) \\ t'_s \in \mathsf{Traces}(\mathsf{Program}(t_s)) \qquad t' = \mathsf{StitchTrace}(t, t'_s, \mathcal{S})}{\mathsf{infer}(\mathsf{SS}, \mathsf{IT}), t \Rightarrow_i t'} \tag{3.1}$$

Soundness and Completeness: Given a trace t, a valid subproblem $\mathcal{S}$, an infered trace t' and $t_s = \mathsf{ExtractTrace}(t, \mathcal{S})$, I prove that my subprogam inference method of extraction and stitching is sound and complete. Soundness in this context means that for all possible mutated subtrace $t'_s \in \mathsf{Traces}(\mathsf{Program}(t_s))$, the stitched trace $t' = \mathsf{StitchTrace}(t, t'_s, \mathcal{S})$ is a valid infered trace. Completeness refers to the fact

$$\begin{tikzcd}
t \arrow[r, "\begin{array}{c} t' \in \mathsf{Traces}(\mathsf{Program}(t)) \\ \mathcal{S} \vdash t \equiv t' \end{array}"] \arrow[d, "t_s = \mathsf{ExtractTrace}(t, \mathcal{S})"'] & t' \\
t, t_s \arrow[r, "t'_s \in \mathsf{Traces}(\mathsf{Program}(t_s))"'] & t, t'_s \arrow[u, "t' = \mathsf{StitchTrace}(t, t'_s, \mathcal{S})"']
\end{tikzcd}$$

Figure 3-3: Entangled vs Independent Subproblem Inference

that for all possible infered traces t', there exists a mutated subtrace t'_s such that $t'_s \in \mathsf{Traces}(\mathsf{Program}(t_s))$ and $t' = \mathsf{StitchTrace}(t, t'_s, \mathcal{S})$.

I summarize the comparison between the entangled subproblem inference and independent subproblem inference approaches in Figure 3-3

I present the theorems, lemmas and their respective proofs to prove the soundness and completeness of my interface below.

3.1 Soundness

Observation 1. *Note that whenever a rule in $\Rightarrow_{ex}$ introduces an* assume *statement in the subtrace, it creates a new variable name. Therefore variable names in the new subtrace do not conflict with any variable names previously introduced in another part of the trace. This fact will be used at various points within this paper.*

Observation 2. *Note that whenever $\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$, $\mathcal{V}(ae) = \mathcal{V}(ae_s)$ and whenever $\mathcal{S} \vdash ae, ae_s, t_s \Rightarrow_{st} ae'$, $\mathcal{V}(ae') = \mathcal{V}(ae_s)$.*

To prove soundness of our interface we start by proving that for a given trace t and a valid subproblem $\mathcal{S}$ on trace t, if for any subtrace t_s the stitching process succeeds, the output trace t' differs from trace t only in parts which are within the subproblem (i.e. $\mathcal{S} \vdash t \equiv t'$).

Formally, for any trace t and subproblem $\mathcal{S}$,

$$t' = \mathsf{StitchTrace}(t, t_s, \mathcal{S}) \implies \mathcal{S} \vdash t \equiv t'$$

Using the definition of $\mathsf{StitchTrace}$, the above lemma can be rewritten as

$$\mathcal{S} \vdash t, \_ \Rightarrow_{st} t' \implies \mathcal{S} \vdash t \equiv t'$$

One will note that the stucture of t_s does not play a significant role in proving the above condition.

To prove the above statement we require a similar condition over augmented expressions embedded within traces. The lemma over augmented expressions is given below:

Lemma 1. *For all augmented expressions ae, ae',*

$$\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_ \implies \mathcal{S} \vdash ae \equiv ae'$$

Proof. Proof using induction.

Base Case:

Case 1: $ae = (x : x)\#id$,

By assumption

$$\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_$$

By definition of $\Rightarrow_{st}$

$$\mathcal{S} \vdash (x : x)\#id', \_, \_ \Rightarrow_{st} (x : x)\#id', \_$$

Then $ae' = (x : x)\#id'$.

By definition of $\equiv$

$$\mathcal{S} \vdash (x : x)\#id \equiv (x : x)\#id'$$

Therefore

$$\mathcal{S} \vdash ae \equiv ae'$$

Therefore when $ae = (x : x)\#id$,

$$\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_ \implies \mathcal{S} \vdash ae \equiv ae'$$

Case 2: $ae = (x(id_v) : v)\#id$

By assumption

$$\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_$$

By definition of $\Rightarrow_{st}$

$$\mathcal{S} \vdash (x(id_v) : v)\#id, \_, \_ \Rightarrow_{st} (x(id'_v) : v')\#id', \_$$

Then $ae' = (x(id'_v) : v')\#id'$.

By definition of $\equiv$

$$\mathcal{S} \vdash (x(id_v) : v)\#id \equiv (x(id'_v) : v')\#id'$$

Therefore

$$\mathcal{S} \vdash ae \equiv ae'$$

Therefore when $ae = (x(id_v) : v)\#id$,

$$\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_ \implies \mathcal{S} \vdash ae \equiv ae'$$

Case 3: $ae = (\lambda.x\ e : v)\#id$

By assumption

$$\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_$$

By definition of $\Rightarrow_{st}$

$$\mathcal{S} \vdash (\lambda.x\ e : v)\#id, \_, \_ \Rightarrow_{st} (\lambda.x\ e : v')\#id', \_$$

Then $ae' = (\lambda.x\ e : v')\#id'$.

By definition of $\equiv$

$$\mathcal{S} \vdash (\lambda.x\ e : v)\#id \equiv (\lambda.x\ e : v')\#id'$$

Therefore

$$\mathcal{S} \vdash ae \equiv ae'$$

Therefore when $ae = (\lambda.x\ e : v)\#id$,

$$\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_ \implies \mathcal{S} \vdash ae \equiv ae'$$

Induction Cases:

Case 1: $ae = ((ae_1\ ae_2) \perp: v)\#id$ and $ID(ae_1) \notin \mathcal{S}$

By assumption

$$\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_$$

By definition of $\Rightarrow_{st}$

$$\mathcal{S} \vdash ((ae_1\ ae_2) \perp: v)\#id, \_, \_ \Rightarrow_{st} ((ae_1'\ ae_2') \perp: v')\#id', \_$$

Then $ae' = ((ae_1\ ae_2) \perp: v')\#id'$, $\mathcal{S} \vdash ae_1, \_, \_ \Rightarrow_{st} ae_1', \_$, and $\mathcal{S} \vdash ae_2, \_, \_ \Rightarrow_{st} ae_2', \_$.

By induction hypothesis

$$\mathcal{S} \vdash ae_1, \_, \_ \Rightarrow_{st} ae_1', \_ \implies \mathcal{S} \vdash ae_1 \equiv ae_1'$$

Because $\mathcal{S} \vdash ae_1, \_, \_ \Rightarrow_{st} ae_1', \_$

$$\mathcal{S} \vdash ae_1 \equiv ae_1'$$

By induction hypothesis

$$\mathcal{S} \vdash ae_2, \_, \_ \Rightarrow_{st} ae'_2, \_ \implies \mathcal{S} \vdash ae_2 \equiv ae'_2$$

Because $\mathcal{S} \vdash ae_2, \_, \_ \Rightarrow_{st} ae'_2, \_$

$$\mathcal{S} \vdash ae_2 \equiv ae'_2$$

Because $\mathcal{S} \vdash ae_1 \equiv ae'_1$ and $\mathcal{S} \vdash ae_2 \equiv ae'_2$, by definition of $\equiv$

$$\mathcal{S} \vdash ((ae_1\ ae_2)\ \bot\colon v)\#id \equiv ((ae'_1\ ae'_2)\ \bot\colon v')\#id'$$

Therefore

$$\mathcal{S} \vdash ae \equiv ae'$$

Therefore when $ae = ((ae_1\ ae_2)\ \bot\colon v)\#id$,

$$\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_ \implies \mathcal{S} \vdash ae \equiv ae'$$

Case 2: $ae = ((ae_1\ ae_2)aa : v)\#id$ and $ID(ae_1) \in \mathcal{S}$

By assumption

$$\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_$$

By definition of $\Rightarrow_{st}$

$$\mathcal{S} \vdash ((ae_1\ ae_2)aa : v)\#id, \_, \_ \Rightarrow_{st} ((ae'_1\ ae'_2)aa' : v')\#id', \_$$

Then $ae' = ((ae_1\ ae_2)aa' : v')\#id'$, $\mathcal{S} \vdash ae_1, \_, \_ \Rightarrow_{st} ae'_1, \_$, and $\mathcal{S} \vdash ae_2, \_, \_ \Rightarrow_{st} ae'_2, \_$.

By induction hypothesis

$$\mathcal{S} \vdash ae_1, \_, \_ \Rightarrow_{st} ae'_1, \_ \implies \mathcal{S} \vdash ae_1 \equiv ae'_1$$

Because $\mathcal{S} \vdash ae_1, \_, \_ \Rightarrow_{st} ae'_1, \_$

$$\mathcal{S} \vdash ae_1 \equiv ae'_1$$

By induction hypothesis

$$\mathcal{S} \vdash ae_2, \_, \_ \Rightarrow_{st} ae'_2, \_ \implies \mathcal{S} \vdash ae_2 \equiv ae'_2$$

Because $\mathcal{S} \vdash ae_2, \_, \_ \Rightarrow_{st} ae'_2, \_$

$$\mathcal{S} \vdash ae_2 \equiv ae'_2$$

Because $\mathcal{S} \vdash ae_1 \equiv ae'_1$ and $\mathcal{S} \vdash ae_2 \equiv ae'_2$, by definition of $\equiv$

$$\mathcal{S} \vdash ((ae_1\ ae_2)aa : v)\#id \equiv ((ae'_1\ ae'_2)aa' : v')\#id'$$

Therefore

$$\mathcal{S} \vdash ae \equiv ae'$$

Therefore when $ae = ((ae_1\ ae_2)aa : v)\#id$,

$$\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_ \implies \mathcal{S} \vdash ae \equiv ae'$$

Case 3: $ae = ((ae_1\ ae_2)x = ae_3 : v)\#id$ and $ID(ae_1) \notin \mathcal{S}$

By assumption

$$\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_$$

By definition of $\Rightarrow_{st}$

$$\mathcal{S} \vdash ((ae_1\ ae_2)x = ae_3 : v)\#id, \_, \_ \Rightarrow_{st} ((ae'_1\ ae'_2)y = ae'_3 : v')\#id', \_$$

Then $ae' = ((ae_1\ ae_2)y = ae'_3 : v')\#id'$, $\mathcal{S} \vdash ae_1, \_, \_ \Rightarrow_{st} ae'_1, \_$, and $\mathcal{S} \vdash ae_2, \_, \_ \Rightarrow_{st} ae'_2, \_$.

By induction hypothesis

$$\mathcal{S} \vdash ae_1, \_, \_ \Rightarrow_{st} ae'_1, \_ \implies \mathcal{S} \vdash ae_1 \equiv ae'_1$$

Because $\mathcal{S} \vdash ae_1, \_, \_ \Rightarrow_{st} ae'_1, \_$

$$\mathcal{S} \vdash ae_1 \equiv ae'_1$$

By induction hypothesis

$$\mathcal{S} \vdash ae_2, \_, \_ \Rightarrow_{st} ae'_2, \_ \implies \mathcal{S} \vdash ae_2 \equiv ae'_2$$

Because $\mathcal{S} \vdash ae_2, \_, \_ \Rightarrow_{st} ae'_2, \_$

$$\mathcal{S} \vdash ae_2 \equiv ae'_2$$

By induction hypothesis

$$\mathcal{S} \vdash ae_3, \_, \_ \Rightarrow_{st} ae'_3, \_ \implies \mathcal{S} \vdash ae_3 \equiv ae'_3$$

Because $\mathcal{S} \vdash ae_3, \_, \_ \Rightarrow_{st} ae'_3, \_$

$$\mathcal{S} \vdash ae_3 \equiv ae'_3$$

Because $\mathcal{S} \vdash ae_1 \equiv ae'_1$, $\mathcal{S} \vdash ae_2 \equiv ae'_2$ and $\mathcal{S} \vdash ae_3 \equiv ae'_3$, by definition of $\equiv$

$$\mathcal{S} \vdash ((ae_1\ ae_2)x = ae_3 : v)\#id \equiv ((ae'_1\ ae'_2)y = ae'_3 : v')\#id'$$

Therefore

$$\mathcal{S} \vdash ae \equiv ae'$$

Therefore when $ae = ((ae_1\ ae_2)x = ae_3 : v)\#id$,

$$\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_ \implies \mathcal{S} \vdash ae \equiv ae'$$

Case 4: $ae = (\mathsf{Dist}(ae_1\#id_e) = ae_2) : v)\#id$ and $id_e \notin \mathcal{S}$.

By assumption

$$\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_$$

By definition of $\Rightarrow_{st}$

$$\mathcal{S} \vdash (\mathsf{Dist}(ae_1\#id_e) = ae_2) : v)\#id, \_, \_ \Rightarrow_{st} (\mathsf{Dist}(ae_1'\#id_e) = ae_2') : v')\#id', \_$$

Then $ae' = (\mathsf{Dist}(ae_1'\#id_e) = ae_2') : v')\#id'$, $\mathcal{S} \vdash ae_1, \_, \_ \Rightarrow_{st} ae_1', \_$, and $\mathcal{S} \vdash ae_2, \_, \_ \Rightarrow_{st} ae_2', \_$.

By induction hypothesis

$$\mathcal{S} \vdash ae_1, \_, \_ \Rightarrow_{st} ae_1', \_ \implies \mathcal{S} \vdash ae_1 \equiv ae_1'$$

Because $\mathcal{S} \vdash ae_1, \_, \_ \Rightarrow_{st} ae_1', \_$

$$\mathcal{S} \vdash ae_1 \equiv ae_1'$$

By induction hypothesis

$$\mathcal{S} \vdash ae_2, \_, \_ \Rightarrow_{st} ae_2', \_ \implies \mathcal{S} \vdash ae_2 \equiv ae_2'$$

Because $\mathcal{S} \vdash ae_2, \_, \_ \Rightarrow_{st} ae_2', \_$

$$\mathcal{S} \vdash ae_2 \equiv ae_2'$$

Because $\mathcal{S} \vdash ae_1 \equiv ae_1'$ and $\mathcal{S} \vdash ae_2 \equiv ae_2'$, by definition of $\equiv$

$$\mathcal{S} \vdash (\mathsf{Dist}(ae_1\#id_e) = ae_2) : v)\#id \equiv (\mathsf{Dist}(ae_1'\#id_e) = ae_2') : v')\#id'$$

Therefore

$$\mathcal{S} \vdash ae \equiv ae'$$

Therefore when $ae = (\mathsf{Dist}(ae_1\#id_e) = ae_2) : v)\#id$ and $id_e \notin \mathcal{S}$,

$$\mathcal{S} \vdash ae, \_, \_ \Rrightarrow_{st} ae', \_ \implies \mathcal{S} \vdash ae \equiv ae'$$

Case 5: $ae = (\mathsf{Dist}(ae_1\#id_e) = ae_2) : v)\#id$ and $id_e \in \mathcal{S}$.

By assumption

$$\mathcal{S} \vdash ae, \_, \_ \Rrightarrow_{st} ae', \_$$

By definition of $\Rrightarrow_{st}$

$$\mathcal{S} \vdash (\mathsf{Dist}(ae_1\#id_e) = ae_2) : v)\#id, \_, \_ \Rrightarrow_{st} (\mathsf{Dist}(ae'_1\#id_e) = ae'_2) : v')\#id', \_$$

Then $ae' = (\mathsf{Dist}(ae'_1\#id_e) = ae'_2) : v')\#id'$ and $\mathcal{S} \vdash ae_1, \_, \_ \Rrightarrow_{st} ae'_1, \_$.

By induction hypothesis

$$\mathcal{S} \vdash ae_1, \_, \_ \Rrightarrow_{st} ae'_1, \_ \implies \mathcal{S} \vdash ae_1 \equiv ae'_1$$

Because $\mathcal{S} \vdash ae_1, \_, \_ \Rrightarrow_{st} ae'_1, \_$

$$\mathcal{S} \vdash ae_1 \equiv ae'_1$$

Because $\mathcal{S} \vdash ae_1 \equiv ae'_1$, by definition of $\equiv$

$$\mathcal{S} \vdash (\mathsf{Dist}(ae_1\#id_e) = ae_2) : v)\#id \equiv (\mathsf{Dist}(ae'_1\#id_e) = ae'_2) : v')\#id'$$

Therefore

$$\mathcal{S} \vdash ae \equiv ae'$$

Therefore when $ae = (\mathsf{Dist}(ae_1\#id_e) = ae_2) : v)\#id$ and $id_e \in \mathcal{S}$,

$$\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_ \implies \mathcal{S} \vdash ae \equiv ae'$$

Because all cases are covered, using induction, the following statement is true for all augmented expressions ae, ae' and subproblems $\mathcal{S}$.

$$\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_ \implies \mathcal{S} \vdash ae \equiv ae'$$

□

Next we use Lemma 1 to prove the lemma below::

Lemma 2. *Given traces t and t' and a subproblem $\mathcal{S}$,*

$$\mathcal{S} \vdash t, \_ \Rightarrow_{st} t' \implies \mathcal{S} \vdash t \equiv t'$$

Proof. Proof by Induction

Base Case: $t = \emptyset$

By assumption

$$\mathcal{S} \vdash t, \_ \Rightarrow_{st} t'$$

By definition of $\Rightarrow_{st}$

$$\mathcal{S} \vdash \emptyset, \_ \Rightarrow_{st} \emptyset$$

Then $t' = \emptyset$.

By definition of $\equiv$

$$\mathcal{S} \vdash \emptyset \equiv \emptyset$$

Therefore

$$\mathcal{S} \vdash t \equiv t'$$

Therefore when $t = \emptyset$

$$\mathcal{S} \vdash t, \_ \Rightarrow_{st} t' \implies \mathcal{S} \vdash t \equiv t'$$

Induction Case:

Case 1: $t = t_s; \textsf{assume}\ x = ae$

By assumption

$$\mathcal{S} \vdash t, \_ \Rightarrow_{st} t'$$

By definition of $\Rightarrow_{st}$

$$\mathcal{S} \vdash t_s; \textsf{assume}\ x = ae, \_ \Rightarrow_{st} t'_s; \textsf{assume}\ x = ae'$$

Then $t' = t'_s; \textsf{assume}\ x = ae'$, $\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_$, and $\mathcal{S} \vdash t_s, \_ \Rightarrow_{st} t'_s$.

By induction hypothesis

$$\mathcal{S} \vdash t_s, \_ \Rightarrow_{st} t'_s \implies \mathcal{S} \vdash t_s \equiv t'_s$$

Because $\mathcal{S} \vdash t_s, \_ \Rightarrow_{st} t'_s$

$$\mathcal{S} \vdash t_s \equiv t'_s$$

From Lemma 1

$$\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_ \implies \mathcal{S} \vdash ae \equiv ae'$$

Because $\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_$

$$\mathcal{S} \vdash ae \equiv ae'$$

Because $\mathcal{S} \vdash t'_s \equiv t_s$, and $\mathcal{S} \vdash ae \equiv ae'$, by definition of $\equiv$

$$\mathcal{S} \vdash t_s; \textsf{assume}\ x = ae \equiv t'_s; \textsf{assume}\ x = ae'$$

Therefore

$$\mathcal{S} \vdash t \equiv t'$$

Therefore when $t = t_s; \textsf{assume}\ x = ae$

$$\mathcal{S} \vdash t, \_ \Rightarrow_{st} t' \implies \mathcal{S} \vdash t \equiv t'$$

Case 2: $t = t_s; \mathsf{observe}(\mathsf{Dist}(ae) = e_v)$

By assumption

$$\mathcal{S} \vdash t, \_ \Rightarrow_{st} t'$$

By definition of $\Rightarrow_{st}$

$$\mathcal{S} \vdash t_s; \mathsf{observe}(\mathsf{Dist}(ae) = e_v), \_ \Rightarrow_{st} t_s'; \mathsf{observe}(\mathsf{Dist}(ae') = e_v)$$

Then $t' = t_s'; \mathsf{observe}(\mathsf{Dist}(ae') = e_v)$, $\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_$, and $\mathcal{S} \vdash t_s, \_ \Rightarrow_{st} t_s'$.

By induction hypothesis

$$\mathcal{S} \vdash t_s, \_ \Rightarrow_{st} t_s' \implies \mathcal{S} \vdash t_s \equiv t_s'$$

Because $\mathcal{S} \vdash t_s, \_ \Rightarrow_{st} t_s'$

$$\mathcal{S} \vdash t_s \equiv t_s'$$

From Lemma 1

$$\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_ \implies \mathcal{S} \vdash ae \equiv ae'$$

Because $\mathcal{S} \vdash ae, \_, \_ \Rightarrow_{st} ae', \_$

$$\mathcal{S} \vdash ae \equiv ae'$$

Because $\mathcal{S} \vdash t_s' \equiv t_s$ and $\mathcal{S} \vdash ae \equiv ae'$, by definition of $\equiv$

$$\mathcal{S} \vdash t_s; \mathsf{observe}(\mathsf{Dist}(ae) = e_v) \equiv t_s'; \mathsf{observe}(\mathsf{Dist}(ae') = e_v)$$

Therefore

$$\mathcal{S} \vdash t \equiv t'$$

Therefore when $t = t_s; \mathsf{observe}(\mathsf{Dist}(ae') = e_v)$

$$\mathcal{S} \vdash t, \_ \Rightarrow_{st} t' \implies \mathcal{S} \vdash t \equiv t'$$

Because all cases have been covered, using induction, for all traces t, t', subproblems $\mathcal{S}$ and subtrace t_s,

$$\mathcal{S} \vdash t, t_s \Rightarrow_{st} t' \implies \mathcal{S} \vdash t \equiv t'$$

□

Corollary 1. *Given a valid trace t, a valid subproblem $\mathcal{S}$, a valid subtrace t_s, for all traces t' :*

$$t' = \mathsf{StitchTrace}(t, t_s, \mathcal{S}) \implies \mathcal{S} \vdash t \equiv t'$$

Therefore, given a trace t and a valid subproblem $\mathcal{S}$ on t, if the stitching process succeeds, then the output trace t' will only differ from trace t with parts which are within the subproblem $\mathcal{S}$.

Next, we prove that given a valid trace t, a valid subproblem $\mathcal{S}$ on trace t, and a subtrace $t_s = \mathsf{ExtractTrace}(t, \mathcal{S})$, for any subtrace $t'_s \in \mathsf{Traces}(\mathsf{Program}(t_s))$, the stitched trace $t' = \mathsf{StitchTrace}(t, t'_s, \mathcal{S})$ is a valid trace from the program of trace t (i.e. $t' \in \mathsf{Traces}(\mathsf{Program}(t))$).

Formally, given a valid trace t, a valid subproblem $\mathcal{S}$ on t, and a subtrace $t_s = \mathsf{ExtractTrace}(t, \mathcal{S})$,

$$\forall\ t'_s \in \mathsf{Traces}(\mathsf{Program}(t_s)).\ t' = \mathsf{StitchTrace}(t, t'_s, \mathcal{S}) \wedge t' \in \mathsf{Traces}(\mathsf{Program}(t))$$

To prove the above statement, we require a few lemmas first which prove a similar condition for augmented expressions and traces under non-empty environment

Lemma 3. *Given environements $\sigma_v, \sigma_{id}, \sigma'_v$ and σ'_{id} such that* $\mathsf{dom}\ \sigma_v = \mathsf{dom}\ \sigma_{id} = \mathsf{dom}\ \sigma'_v = \mathsf{dom}\ \sigma'_{id}$ *an augmented expression ae within a trace t, and a valid subproblem $\mathcal{S}$ over trace t*

$$\exists\ e, p_s.\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae \wedge \mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

$$\wedge t_s; \mathsf{assume}\ z = ae_s \Rightarrow_r p_s \wedge \sigma'_v, \sigma'_{id} \vdash p_s \Rightarrow_s t'_s; \mathsf{assume}\ z = ae'_s$$

$$\implies \exists\ ae'.\mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae', \_ \wedge \sigma'_v, \sigma'_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

Proof. Proof by induction

Base Case:

Case 1: $ae = (x : x)\#id$

By assumption

$$\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash x \Rightarrow_s \_, \_, (x : x)\#id$$

Then $e = x$ and $x \notin \textsf{dom}\ \sigma_v$.

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash (x : x)\#id \Rightarrow_{ex} (x : x)\#id, \emptyset$$

Then $ae_s = (x : x)\#id$ and $t_s = \emptyset$.

By assumption

$$t_s; \textsf{assume}\ z = ae_s \Rightarrow_r p_s$$

By definition of $\Rightarrow_r$

$$\textsf{assume}\ z = (x : x)\#id \Rightarrow_r \textsf{assume}\ z = x$$

Then $p_s = \textsf{assume}\ z = x$.

By assumption

$$\sigma'_v, \sigma'_{id} \vdash p_s \Rightarrow_s t'_s; \textsf{assume}\ z = ae'_s$$

By definition of $\Rightarrow_s$, $\textsf{dom}\ \sigma'_v = \textsf{dom}\ \sigma_v$

$$\sigma'_v, \sigma'_{id} \vdash \textsf{assume}\ z = x \Rightarrow_s \textsf{assume}\ z = (x : x)\#id'$$

Then $t'_s = \emptyset$ and $ae'_s = (x : x)\#id'$.

Consider $ae' = (x : x)\#id'$.

By definition of $\Rightarrow_{st}$

$$\mathcal{S} \vdash (x : x)\#id, (x : x)\#id', \emptyset \Rightarrow_{st} (x : x)\#id'$$

Therefore

$$\mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae'$$

By definition of $\Rightarrow_s$ and $x \in \textsf{dom}\ \sigma'_v$

$$\sigma'_v, \sigma'_{id} \vdash x \Rightarrow_s \_, \_, (x : x)\#id'$$

Therefore

$$\sigma'_v, \sigma'_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

Therefore when $ae = (x : x)\#id$

$$\exists\ e, p_s.\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae \land \mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

$$\land t_s; \textsf{assume}\ z = ae_s \Rightarrow_r p_s \land \sigma'_v, \sigma'_{id} \vdash p_s \Rightarrow_s t'_s; \textsf{assume}\ z = ae'_s$$

$$\implies \exists\ ae'.\mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae', \_ \land \sigma'_v, \sigma'_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

Case 2: $ae = (x(id_v) : v)\#id$

By assumption

$$\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash x \Rightarrow_s \_, \_, (x(id_v) : v)\#id$$

Then $e = x$, $v = \sigma_v(x)$, and $id_v = \sigma_{id}(x)$.

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash (x(id_v) : v)\#id \Rightarrow_{ex} (x(id_v) : v)\#id, \emptyset$$

Then $ae_s = (x(id_v) : v)\#id$ and $t_s = \emptyset$.

By assumption

$$t_s; \textsf{assume}\ z = ae_s \Rightarrow_r p_s$$

By definition of $\Rightarrow_r$

$$\textsf{assume}\ z = (x(id_v) : v)\#id \Rightarrow_r \textsf{assume}\ z = x$$

Then $p_s = \textsf{assume}\ z = x$.

By assumption

$$\sigma'_v, \sigma'_{id} \vdash p_s \Rightarrow_s t'_s; \textsf{assume}\ z = ae'_s$$

By definition of $\Rightarrow_s$, $\textsf{dom}\ \sigma'_v = \textsf{dom}\ \sigma_v$

$$\sigma'_v, \sigma'_{id} \vdash \textsf{assume}\ z = x \Rightarrow_s \textsf{assume}\ z = (x(id'_v) : v')\#id'$$

Then $t'_s = \emptyset$, $ae'_s = (x(id'_v) : v')\#id'$, $v' = \sigma'_v(x)$, and $id'_v = \sigma'_{id}(x)$.

Consider $ae' = (x(id'_v) : v')\#id'$.

By definition of $\Rightarrow_{st}$

$$\mathcal{S} \vdash (x(id_v) : v)\#id, (x(id'_v) : v')\#id', \emptyset \Rightarrow_{st} (x(id'_v) : v')\#id'$$

Therefore

$$\mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae'$$

By definition of $\Rightarrow_s$,$v' = \sigma'_v(x)$, and $id'_v = \sigma'_{id}(x)$

$$\sigma'_v, \sigma'_{id} \vdash x \Rightarrow_s \_, \_, (x(id'_v) : v')\#id'$$

Therefore

$$\sigma'_v, \sigma'_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

Therefore when $ae = (x(id_v) : v)\#id$

$$\exists\ e, p_s.\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae \land \mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

$$\land t_s; \textsf{assume } z = ae_s \Rightarrow_r p_s \land \sigma'_v, \sigma'_{id} \vdash p_s \Rightarrow_s t'_s; \textsf{assume } z = ae'_s$$

$$\implies \exists\ ae'.\mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae', \_ \land \sigma'_v, \sigma'_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

Case 3: $ae = (\lambda.x\ e' : v)\#id$

By assumption

$$\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash \lambda.x\ e' \Rightarrow_s \_, \_, (\lambda.x\ e' : v)\#id$$

Then $e = \lambda.x\ e'$.

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash (\lambda.x\ e' : v)\#id \Rightarrow_{ex} (\lambda.x\ e' : v)\#id, \emptyset$$

Then $ae_s = (\lambda.x\ e' : v)\#id$ and $t_s = \emptyset$.

By assumption

$$t_s; \textsf{assume } z = ae_s \Rightarrow_r p_s$$

By definition of $\Rightarrow_r$

$$\textsf{assume } z = (\lambda.x\ e' : v)\#id \Rightarrow_r \textsf{assume } z = \lambda.x\ e'$$

Then $p_s = \textsf{assume}\ z = \lambda.x\ e'$.

By assumption

$$\sigma'_v, \sigma'_{id} \vdash p_s \Rightarrow_s t'_s; \textsf{assume}\ z = ae'_s$$

By definition of $\Rightarrow_s$, $\textsf{dom}\ \sigma'_v = \textsf{dom}\ \sigma_v$

$$\sigma'_v, \sigma'_{id} \vdash \textsf{assume}\ z = \lambda.x\ e' \Rightarrow_s \textsf{assume}\ z = (\lambda.x\ e' : v')\#id'$$

Then $t'_s = \emptyset$ and $ae'_s = (\lambda.x\ e' : v')\#id'$.

Consider $ae' = (\lambda.x\ e' : v')\#id'$.

By definition of $\Rightarrow_{st}$

$$\mathcal{S} \vdash (\lambda.x\ e' : v)\#id, (\lambda.x\ e' : v')\#id', \emptyset \Rightarrow_{st} (\lambda.x\ e' : v')\#id'$$

Therefore

$$\mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae'$$

By definition of $\Rightarrow_s$ and $\sigma'_v, \sigma'_{id} \vdash \lambda.x\ e' \Rightarrow_s \_, \_, (\lambda.x\ e' : v')\#id'$

$$\sigma'_v, \sigma'_{id} \vdash \lambda.x\ e' \Rightarrow_s \_, \_, (\lambda.x\ e' : v')\#id'$$

Therefore

$$\sigma'_v, \sigma'_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

Therefore when $ae = (\lambda.x\ e' : v)\#id$

$$\exists\ e, p_s.\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae \land \mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

$$\land t_s; \textsf{assume}\ z = ae_s \Rightarrow_r p_s \land \sigma'_v, \sigma'_{id} \vdash p_s \Rightarrow_s t'_s; \textsf{assume}\ z = ae'_s$$

$$\implies \exists\ ae'.\mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae', \_ \land \sigma'_v, \sigma'_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

Induction Cases:

Case 1: $ae = (\textsf{Dist}(ae_1\#id_e) = ae_v : v)\#id$ and $id_e \in \mathcal{S}$

By assumption

$$\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash \mathsf{Dist}(e_1) \Rightarrow_s \_, \_, (\mathsf{Dist}(ae_1\#id_e) = ae_v : v)\#id$$

Then $e = \mathsf{Dist}(e_1)$, $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1$.

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash (\mathsf{Dist}(ae_1\#id_e) = ae_v : v)\#id \Rightarrow_{ex} (\mathsf{Dist}(ae_s^1\#id_e) = ae_v : v)\#id, t_s^1$$

Then $ae_s = (\mathsf{Dist}(ae_1\#id_e) = ae_v : v)\#id$, $t_s = t_s^1$, and $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$.

By assumption

$$t_s; \mathsf{assume}\ z = ae_s \Rightarrow_r p_s$$

By definition of $\Rightarrow_r$

$$\mathsf{assume}\ z = (\mathsf{Dist}(ae_s^1\#id_e) = ae_v : v)\#id \Rightarrow_r \mathsf{assume}\ z = \mathsf{Dist}(e_s^1)$$

Then $p_s = p_s^1; \mathsf{assume}\ z = \mathsf{Dist}(e_s^1)$ and $ae_s^1 \Rightarrow_r e_s^1$.

By assumption

$$\sigma_v', \sigma_{id}' \vdash p_s \Rightarrow_s t_s'; \mathsf{assume}\ z = ae_s'$$

By definition of $\Rightarrow_s$,

$$\sigma_v', \sigma_{id}' \vdash p_s^1; \mathsf{assume}\ z = \mathsf{Dist}(e_1) \Rightarrow_s t_s^2; \mathsf{assume}\ z = (\mathsf{Dist}(ae_s^2\#id_e') = ae_v' : v')\#id'$$

Then $t_s' = t_s^2$, $ae_s' = (\mathsf{Dist}(ae_s^2\#id_e) = ae_v' : v')\#id'$, and $\sigma_v', \sigma_{id}' \vdash p_s^1; \mathsf{assume}\ z = e_1 \Rightarrow_s t_s^2; \mathsf{assume}\ z = ae_s^2$.

By induction hypothesis

$$\exists\ e_1, p_s^1, e_s^1.\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1 \wedge \mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$$

$$\wedge t_s^1; \textsf{assume } z = ae_s^1 \Rightarrow_r p_s^1; \textsf{assume } z = e_s^1$$

$$\wedge \sigma_v', \sigma_{id}' \vdash p_s^1; \textsf{assume } z = e_s^1 \Rightarrow_s t_s^2; \textsf{assume } z = ae_s^2$$

$$\Longrightarrow\ \exists\ ae_1'.\mathcal{S} \vdash ae_1, ae_s^2, t_s^2 \Rightarrow_{st} ae_1' \wedge \sigma_v', \sigma_{id}' \vdash e_1 \Rightarrow_s \_, \_, ae_1'$$

Because $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1$, $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$, $t_s^1; \textsf{assume } z = ae_s^1 \Rightarrow_r p_s^1; \textsf{assume } z = e_s^1$, and $\sigma_v', \sigma_{id}' \vdash p_s^1; \textsf{assume } z = e_s^1 \Rightarrow_s t_s^2; \textsf{assume } z = ae_s^2$.

$$\mathcal{S} \vdash ae_1, ae_s^2, t_s^2 \Rightarrow_{st} ae_1' \wedge \sigma_v', \sigma_{id}' \vdash e_1 \Rightarrow_s \_, \_, ae_1'$$

Consider $ae' = (\textsf{Dist}(ae_1'\#id_e') = ae_v' : v')\#id'$.

By definition of $\Rightarrow_{st}$, $\mathcal{S} \vdash ae_1, ae_s^2, t_s^2 \Rightarrow_{st} ae_1'$

$$\mathcal{S} \vdash (\textsf{Dist}(ae_s^1\#id_e) = ae_v : v)\#id, (\textsf{Dist}(ae_s^2\#id_e') = ae_v' : v')\#id', t_s' \\ \Rightarrow_{st} (\textsf{Dist}(ae_1'\#id_e') = ae_v' : v')\#id'$$

Therefore

$$\mathcal{S} \vdash ae, ae_s', t_s' \Rightarrow_{st} ae'$$

By definition of $\Rightarrow_s$ and $\sigma_v', \sigma_{id}' \vdash e_1 \Rightarrow_s \_, \_, ae_1$,

$$\sigma_v', \sigma_{id}' \vdash e \Rightarrow_s \_, \_, (\textsf{Dist}(ae_1'\#id_e') = ae_v' : v')\#id'$$

Therefore

$$\sigma_v', \sigma_{id}' \vdash e \Rightarrow_s \_, \_, ae'$$

Therefore when $ae = (\textsf{Dist}(ae_1\#id_e) = ae_v : v)\#id$

$$\exists\ e, p_s.\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae \wedge \mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

$$\wedge t_s; \textsf{assume } z = ae_s \Rightarrow_r p_s \wedge \sigma'_v, \sigma'_{id} \vdash p_s \Rightarrow_s t'_s; \textsf{assume } z = ae'_s$$

$$\Longrightarrow \exists\, ae'.\mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae', \_ \wedge \sigma'_v, \sigma'_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

Case 2: $ae = (\textsf{Dist}(ae_1\#id_e) = ae_2 : v)\#id$ and $id_e \notin \mathcal{S}$

By assumption

$$\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash \textsf{Dist}(e_1) \Rightarrow_s \_, \_, (\textsf{Dist}(ae_1\#id_e) = ae_2 : v)\#id$$

Then $e = \textsf{Dist}(e_1)$, $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1$, $ae_2 \Rightarrow_r e_2$ and $\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae_2$.

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash (\textsf{Dist}(ae_1\#id_e) = ae_2 : v)\#id \Rightarrow_{ex} ae^3_s, t^1_s; \textsf{observe}(\textsf{Dist}(ae^1_s) = e_2); t^3_s$$

Then $ae_s = ae^3_s$, $t_s = t^1_s; \textsf{observe}(\textsf{Dist}(ae^1_s) = e_2); t^3_s$, $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae^1_s, t^1_s$, and $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae^3_s, t^3_s$.

By assumption

$$t_s; \textsf{assume } z = ae_s \Rightarrow_r p_s$$

By definition of $\Rightarrow_r$

$$\begin{gathered} t^1_s; \textsf{observe}(\textsf{Dist}(ae^1_s) = e_2); t^3_s; \textsf{assume } z = ae^3_s \Rightarrow_r \\ p^1_s; \textsf{observe}(\textsf{Dist}(e^1_s) = e_2); p^2_s\textsf{assume } z = \textsf{Dist}(e^1_s) \end{gathered}$$

Then $p_s = p^1_s; \textsf{observe}(\textsf{Dist}(e^1_s) = e_2); p^2_s; \textsf{assume } z = e^2_s$, $ae^1_s \Rightarrow_r e^1_s$, $t^1_s \Rightarrow_r p^1_s$, $ae^3_s \Rightarrow_r e^2_s$, and $t^3_s \Rightarrow_r p^2_s$.

By assumption

$$\sigma'_v, \sigma'_{id} \vdash p_s \Rightarrow_s t'_s; \textsf{assume } z = ae'_s$$

By definition of $\Rightarrow_s$,

$$\sigma'_v, \sigma'_{id} \vdash p^1_s; \mathsf{observe}(\mathsf{Dist}(e^1_s) = e_2); p^2_s; \mathsf{assume}\ z = e^2_s \Rightarrow_s$$
$$t^2_s; \mathsf{observe}(\mathsf{Dist}(ae^2_s) = e_2); t^4_s; \mathsf{assume}\ z = ae^4_s$$

Then $t'_s = t^2_s; \mathsf{observe}(\mathsf{Dist}(ae^2_s) = e_2); t^4_s$, $ae'_s = ae^4_s$, $\sigma'_v, \sigma'_{id} \vdash p^1_s; \mathsf{assume}\ z = e^1_s \Rightarrow_s t^2_s; \mathsf{assume}\ z = ae^2_s$, and $\sigma'_v, \sigma'_{id} \vdash p^2_s; \mathsf{assume}\ z = e^2_s \Rightarrow_s t^4_s; \mathsf{assume}\ z = ae^4_s$.

By induction hypothesis

$$\exists\ e_1, p^1_s, e^1_s.\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1 \land \mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae^1_s, t^1_s$$

$$\land t^1_s; \mathsf{assume}\ z = ae^1_s \Rightarrow_r p^1_s; \mathsf{assume}\ z = e^1_s$$

$$\land \sigma'_v, \sigma'_{id} \vdash p^1_s; \mathsf{assume}\ z = e^1_s \Rightarrow_s t^2_s; \mathsf{assume}\ z = ae^2_s$$

$$\implies \exists\ ae'_1.\mathcal{S} \vdash ae_1, ae^2_s, t^2_s \Rightarrow_{st} ae'_1 \land \sigma'_v, \sigma'_{id} \vdash e_1 \Rightarrow_s \_, \_, ae'_1$$

Because $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1$, $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae^1_s, t^1_s$, $t^1_s; \mathsf{assume}\ z = ae^1_s \Rightarrow_r p^1_s; \mathsf{assume}\ z = e^1_s$, and $\sigma'_v, \sigma'_{id} \vdash p^1_s; \mathsf{assume}\ z = e^1_s \Rightarrow_s t^2_s; \mathsf{assume}\ z = ae^2_s$.

$$\mathcal{S} \vdash ae_1, ae^2_s, t^2_s \Rightarrow_{st} ae'_1 \land \sigma'_v, \sigma'_{id} \vdash e_1 \Rightarrow_s \_, \_, ae'_1$$

By induction hypothesis

$$\exists\ e_2, p^2_s, e^2_s.\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae_2 \land \mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae^3_s, t^3_s$$

$$\land t^3_s; \mathsf{assume}\ z = ae^3_s \Rightarrow_r p^2_s; \mathsf{assume}\ z = e^2_s$$

$$\land \sigma'_v, \sigma'_{id} \vdash p^2_s; \mathsf{assume}\ z = e^2_s \Rightarrow_s t^4_s; \mathsf{assume}\ z = ae^4_s$$

$$\implies \exists\ ae'_2.\mathcal{S} \vdash ae_2, ae^4_s, t^4_s \Rightarrow_{st} ae'_2 \land \sigma'_v, \sigma'_{id} \vdash e_2 \Rightarrow_s \_, \_, ae'_2$$

Because $\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae_2$, $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae^3_s, t^3_s$, $t^3_s; \mathsf{assume}\ z = ae^3_s \Rightarrow_r$

$p_s^2; \mathsf{assume}\ z = e_s^2$, and $\sigma_v', \sigma_{id}' \vdash p_s^2; \mathsf{assume}\ z = e_s^2 \Rightarrow_s t_s^4; \mathsf{assume}\ z = ae_s^4$.

$$\mathcal{S} \vdash ae_2, ae_s^4, t_s^4 \Rightarrow_{st} ae_2' \wedge \sigma_v', \sigma_{id}' \vdash e_2 \Rightarrow_s \_, \_, ae_2'$$

Consider $ae' = (\mathsf{Dist}(ae_1'\#id_e) = ae_2' : v')\#id'$.
By definition of $\Rightarrow_{st}$, $\mathcal{S} \vdash ae_1, ae_s^2, t_s^2 \Rightarrow_{st} ae_1'$, and $\mathcal{S} \vdash ae_2, ae_s^4, t_s^4 \Rightarrow_{st} ae_2'$,

$$\mathcal{S} \vdash (\mathsf{Dist}(ae_1\#id_e) = ae_2 : v)\#id, ae_s^4, t_s' \\ \Rightarrow_{st} (\mathsf{Dist}(ae_1'\#id_e') = ae_2' : v')\#id'$$

Therefore

$$\mathcal{S} \vdash ae, ae_s', t_s' \Rightarrow_{st} ae'$$

By definition of $\Rightarrow_s$, $\sigma_v', \sigma_{id}' \vdash e_1 \Rightarrow_s \_, \_, ae_1'$, $\sigma_v', \sigma_{id}' \vdash e_2 \Rightarrow_s, \_, \_, ae_2'$,

$$\sigma_v', \sigma_{id}' \vdash e \Rightarrow_s \_, \_, (\mathsf{Dist}(ae_1'\#id_e) = ae_2' : v')\#id'$$

Therefore

$$\sigma_v', \sigma_{id}' \vdash e \Rightarrow_s \_, \_, ae'$$

Therefore when $ae = (\mathsf{Dist}(ae_1\#id_e) = ae_2 : v)\#id$

$$\exists\ e, p_s.\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae \wedge \mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

$$\wedge t_s; \mathsf{assume}\ z = ae_s \Rightarrow_r p_s \wedge \sigma_v', \sigma_{id}' \vdash p_s \Rightarrow_s t_s'; \mathsf{assume}\ z = ae_s'$$

$$\implies \exists\ ae'.\mathcal{S} \vdash ae, ae_s', t_s' \Rightarrow_{st} ae', \_ \wedge \sigma_v', \sigma_{id}' \vdash e \Rightarrow_s \_, \_, ae'$$

Case 3: $ae = ((ae_1\ ae_2) \perp: v)\#id$ and $ID(ae_1) \notin \mathcal{S}$
By assumption

$$\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash (e_1\ e_2) \Rightarrow_s \_, \_, ((ae_1\ ae_2) \perp: v)\#id$$

Then $e = (e_1\ e_2)$, $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1$, and $\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae_2$.

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash ((ae_1\ ae_2) \perp\colon v)\#id \Rightarrow_{ex} ((ae_s^1\ ae_s^3) \perp\colon v)\#id, t_s^1; t_s^3$$

Then $ae_s = ((ae_s^1\ ae_s^3) \perp\colon v)\#id$, $t_s = t_s^1; t_s^3$, $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$, and $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^3, t_s^3$.

By assumption

$$t_s; \textsf{assume } z = ae_s \Rightarrow_r p_s$$

By definition of $\Rightarrow_r$

$$t_s^1; t_s^3; \textsf{assume } z = ((ae_s^1\ ae_s^3) \perp\colon v)\#id \Rightarrow_r p_s^1; p_s^2 \textsf{assume } z = (e_s^1\ e_s^2)$$

Then $p_s = p_s^1; p_s^2; \textsf{assume } z = (e_s^1\ e_s^2)$, $ae_s^1 \Rightarrow_r e_s^1$, $t_s^1 \Rightarrow_r p_s^1$, $ae_s^3 \Rightarrow_r e_s^2$, and $t_s^3 \Rightarrow_r p_s^2$.

By assumption

$$\sigma_v', \sigma_{id}' \vdash p_s \Rightarrow_s t_s'; \textsf{assume } z = ae_s'$$

By definition of $\Rightarrow_s$,

$$\sigma_v', \sigma_{id}' \vdash p_s^1; p_s^2; \textsf{assume } z = (e_s^1\ e_s^2) \Rightarrow_s t_s^2; t_s^4; \textsf{assume } z = ((ae_s^2\ ae_s^4) \perp\colon v)\#id$$

Then $t_s' = t_s^2; t_s^4$, $ae_s' = ((ae_s^2\ ae_s^4) \perp\colon v)\#id$, $\sigma_v', \sigma_{id}' \vdash p_s^1; \textsf{assume } z = e_s^1 \Rightarrow_s t_s^2; \textsf{assume } z = ae_s^2$, and $\sigma_v', \sigma_{id}' \vdash p_s^2; \textsf{assume } z = e_s^2 \Rightarrow_s t_s^4; \textsf{assume } z = ae_s^4$.

By induction hypothesis

$$\exists\ e_1, p_s^1, e_s^1 . \sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1 \wedge \mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$$

$$\wedge t_s^1; \textsf{assume } z = ae_s^1 \Rightarrow_r p_s^1; \textsf{assume } z = e_s^1$$

$$\wedge \sigma'_v, \sigma'_{id} \vdash p^1_s; \textsf{assume } z = e^1_s \Rightarrow_s t^2_s; \textsf{assume } z = ae^2_s$$

$$\Longrightarrow \exists\ ae'_1 . \mathcal{S} \vdash ae_1, ae^2_s, t^2_s \Rightarrow_{st} ae'_1 \wedge \sigma'_v, \sigma'_{id} \vdash e_1 \Rightarrow_s \_, \_, ae'_1$$

Because $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1$, $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae^1_s, t^1_s$, $t^1_s; \textsf{assume } z = ae^1_s \Rightarrow_r p^1_s; \textsf{assume } z = e^1_s$, and $\sigma'_v, \sigma'_{id} \vdash p^1_s; \textsf{assume } z = e^1_s \Rightarrow_s t^2_s; \textsf{assume } z = ae^2_s$.

$$\mathcal{S} \vdash ae_1, ae^2_s, t^2_s \Rightarrow_{st} ae'_1 \wedge \sigma'_v, \sigma'_{id} \vdash e_1 \Rightarrow_s \_, \_, ae'_1$$

By induction hypothesis

$$\exists\ e_2, p^2_s, e^2_s . \sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae_2 \wedge \mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae^3_s, t^3_s$$

$$\wedge t^3_s; \textsf{assume } z = ae^3_s \Rightarrow_r p^2_s; \textsf{assume } z = e^2_s$$

$$\wedge \sigma'_v, \sigma'_{id} \vdash p^2_s; \textsf{assume } z = e^2_s \Rightarrow_s t^4_s; \textsf{assume } z = ae^4_s$$

$$\Longrightarrow \exists\ ae'_2 . \mathcal{S} \vdash ae_2, ae^4_s, t^4_s \Rightarrow_{st} ae'_2 \wedge \sigma'_v, \sigma'_{id} \vdash e_2 \Rightarrow_s \_, \_, ae'_2$$

Because $\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae_2$, $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae^3_s, t^3_s$, $t^3_s; \textsf{assume } z = ae^3_s \Rightarrow_r p^2_s; \textsf{assume } z = e^2_s$, and $\sigma'_v, \sigma'_{id} \vdash p^2_s; \textsf{assume } z = e^2_s \Rightarrow_s t^4_s; \textsf{assume } z = ae^4_s$.

$$\mathcal{S} \vdash ae_2, ae^4_s, t^4_s \Rightarrow_{st} ae'_2 \wedge \sigma'_v, \sigma'_{id} \vdash e_2 \Rightarrow_s \_, \_, ae'_2$$

Consider $ae' = ((ae'_1\ ae'_2) \perp\!: v')\#id'$.

By definition of $\Rightarrow_{st}$, $\mathcal{S} \vdash ae_1, ae^2_s, t^2_s \Rightarrow_{st} ae'_1$, and $\mathcal{S} \vdash ae_2, ae^4_s, t^4_s \Rightarrow_{st} ae'_2$,

$$\begin{gathered} \mathcal{S} \vdash ((ae_1\ ae_2) \perp\!: v)\#id, ((ae^2_s\ ae^4_s) \perp\!: v')\#id', t'_s \\ \Rightarrow_{st} ((ae'_1\ ae'_2) \perp\!: v')\#id' \end{gathered}$$

Therefore

$$\mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae'$$

By definition of $\Rightarrow_s$, $\sigma'_v, \sigma'_{id} \vdash e_1 \Rightarrow_s \_, \_, ae'_1$, $\sigma'_v, \sigma'_{id} \vdash e_2 \Rightarrow_s, \_, \_, ae'_2$, and because

ae_1 is not in the subproblem $\mathcal{S}$, therefore its value will not change

$$\sigma'_v, \sigma'_{id} \vdash e \Rightarrow_s \_, \_, ((ae'_1\ ae'_2)\ \bot\colon v')\#id'$$

Therefore

$$\sigma'_v, \sigma'_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

Therefore when $ae = ((ae_1\ ae_2)\ \bot\colon v)\#id$, $ID(ae_1) \notin \mathcal{S}$

$$\exists\ e, p_s.\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae \wedge \mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

$$\wedge t_s; \textsf{assume } z = ae_s \Rightarrow_r p_s \wedge \sigma'_v, \sigma'_{id} \vdash p_s \Rightarrow_s t'_s; \textsf{assume } z = ae'_s$$

$$\implies \exists\ ae'.\mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae', \_ \wedge \sigma'_v, \sigma'_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

Case 4: $ae = ((ae_1\ ae_2)aa : v)\#id$ and $ID(ae_1) \in \mathcal{S}$

By assumption

$$\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash (e_1\ e_2) \Rightarrow_s \_, \_, ((ae_1\ ae_2)aa : v)\#id$$

Then $e = (e_1\ e_2)$, $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1$, and $\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae_2$.

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash ((ae_1\ ae_2)aa : v)\#id \Rightarrow_{ex} ((ae^1_s\ ae^3_s)aa : v)\#id, t^1_s; t^3_s$$

Then $ae_s = ((ae^1_s\ ae^3_s)aa : v)\#id$, $t_s = t^1_s; t^3_s$, $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae^1_s, t^1_s$, and $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae^3_s, t^3_s$.

By assumption

$$t_s; \textsf{assume } z = ae_s \Rightarrow_r p_s$$

By definition of $\Rightarrow_r$

$$t_s^1; t_s^3; \textsf{assume } z = ((ae_s^1\ ae_s^3)aa : v)\#id \Rightarrow_r p_s^1; p_s^2 \textsf{assume } z = (e_s^1\ e_s^2)$$

Then $p_s = p_s^1; p_s^2; \textsf{assume } z = (e_s^1\ e_s^2)$, $ae_s^1 \Rightarrow_r e_s^1$, $t_s^1 \Rightarrow_r p_s^1$, $ae_s^3 \Rightarrow_r e_s^2$, and $t_s^3 \Rightarrow_r p_s^2$.

By assumption

$$\sigma_v', \sigma_{id}' \vdash p_s \Rightarrow_s t_s'; \textsf{assume } z = ae_s'$$

By definition of $\Rightarrow_s$,

$$\sigma_v', \sigma_{id}' \vdash p_s^1; p_s^2; \textsf{assume } z = (e_s^1\ e_s^2) \Rightarrow_s t_s^2; t_s^4; \textsf{assume } z = ((ae_s^2\ ae_s^4)aa' : v)\#id$$

Then $t_s' = t_s^2; t_s^4$, $ae_s' = ((ae_s^2\ ae_s^4)aa' : v)\#id$, $\sigma_v', \sigma_{id}' \vdash p_s^1; \textsf{assume } z = e_s^1 \Rightarrow_s t_s^2; \textsf{assume } z = ae_s^2$, and $\sigma_v', \sigma_{id}' \vdash p_s^2; \textsf{assume } z = e_s^2 \Rightarrow_s t_s^4; \textsf{assume } z = ae_s^4$.

By induction hypothesis

$$\exists\ e_1, p_s^1, e_s^1 . \sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1 \wedge \mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$$

$$\wedge t_s^1; \textsf{assume } z = ae_s^1 \Rightarrow_r p_s^1; \textsf{assume } z = e_s^1$$

$$\wedge \sigma_v', \sigma_{id}' \vdash p_s^1; \textsf{assume } z = e_s^1 \Rightarrow_s t_s^2; \textsf{assume } z = ae_s^2$$

$$\Longrightarrow \exists\ ae_1' . \mathcal{S} \vdash ae_1, ae_s^2, t_s^2 \Rightarrow_{st} ae_1' \wedge \sigma_v', \sigma_{id}' \vdash e_1 \Rightarrow_s \_, \_, ae_1'$$

Because $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1$, $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$, $t_s^1; \textsf{assume } z = ae_s^1 \Rightarrow_r p_s^1; \textsf{assume } z = e_s^1$, and $\sigma_v', \sigma_{id}' \vdash p_s^1; \textsf{assume } z = e_s^1 \Rightarrow_s t_s^2; \textsf{assume } z = ae_s^2$.

$$\mathcal{S} \vdash ae_1, ae_s^2, t_s^2 \Rightarrow_{st} ae_1' \wedge \sigma_v', \sigma_{id}' \vdash e_1 \Rightarrow_s \_, \_, ae_1'$$

By induction hypothesis

$$\exists\ e_2, p_s^2, e_s^2.\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae_2 \wedge \mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^3, t_s^3$$

$$\wedge t_s^3; \textsf{assume } z = ae_s^3 \Rightarrow_r p_s^2; \textsf{assume } z = e_s^2$$

$$\wedge \sigma_v', \sigma_{id}' \vdash p_s^2; \textsf{assume } z = e_s^2 \Rightarrow_s t_s^4; \textsf{assume } z = ae_s^4$$

$$\Longrightarrow\ \exists\ ae_2'.\mathcal{S} \vdash ae_2, ae_s^4, t_s^4 \Rightarrow_{st} ae_2' \wedge \sigma_v', \sigma_{id}' \vdash e_2 \Rightarrow_s \_, \_, ae_2'$$

Because $\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae_2$, $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^3, t_s^3$, $t_s^3; \textsf{assume } z = ae_s^3 \Rightarrow_r p_s^2; \textsf{assume } z = e_s^2$, and $\sigma_v', \sigma_{id}' \vdash p_s^2; \textsf{assume } z = e_s^2 \Rightarrow_s t_s^4; \textsf{assume } z = ae_s^4$.

$$\mathcal{S} \vdash ae_2, ae_s^4, t_s^4 \Rightarrow_{st} ae_2' \wedge \sigma_v', \sigma_{id}' \vdash e_2 \Rightarrow_s \_, \_, ae_2'$$

Consider $ae' = ((ae_1'\ ae_2')aa' : v')\#id'$.
By definition of $\Rightarrow_{st}$, $\mathcal{S} \vdash ae_1, ae_s^2, t_s^2 \Rightarrow_{st} ae_1'$, and $\mathcal{S} \vdash ae_2, ae_s^4, t_s^4 \Rightarrow_{st} ae_2'$,

$$\begin{gathered}\mathcal{S} \vdash ((ae_1\ ae_2)aa : v)\#id, ((ae_s^2\ ae_s^4)aa' : v')\#id', t_s' \\ \Rightarrow_{st} ((ae_1'\ ae_2')aa' : v')\#id'\end{gathered}$$

Therefore

$$\mathcal{S} \vdash ae, ae_s', t_s' \Rightarrow_{st} ae'$$

By definition of $\Rightarrow_s$, $\sigma_v', \sigma_{id}' \vdash e_1 \Rightarrow_s \_, \_, ae_1'$, $\sigma_v', \sigma_{id}' \vdash e_2 \Rightarrow_s, \_, \_, ae_2'$, and because $\mathcal{V}(ae_1') = \mathcal{V}(ae_s^2)$, $\mathcal{V}(ae_2') = \mathcal{V}(ae_s^4)$ (Observation 2)

$$\sigma_v', \sigma_{id}' \vdash e \Rightarrow_s \_, \_, ((ae_1'\ ae_2')aa' : v')\#id'$$

Therefore

$$\sigma_v', \sigma_{id}' \vdash e \Rightarrow_s \_, \_, ae'$$

Therefore when $ae = ((ae_1\ ae_2)aa : v)\#id$, $ID(ae_1) \in \mathcal{S}$

$$\exists\ e, p_s.\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae \wedge \mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

$$\wedge t_s; \textsf{assume } z = ae_s \Rightarrow_r p_s \wedge \sigma_v', \sigma_{id}' \vdash p_s \Rightarrow_s t_s'; \textsf{assume } z = ae_s'$$

$$\implies \exists\, ae'.\mathcal{S} \vdash ae, ae_s', t_s' \Rightarrow_{st} ae', \_ \wedge \sigma_v', \sigma_{id}' \vdash e \Rightarrow_s \_, \_, ae'$$

Case 5: $ae = ((ae_1\ ae_2)y = ae_3 : v)\#id$ and $ID(ae_1) \notin \mathcal{S}$

By assumption

$$\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash (e_1\ e_2) \Rightarrow_s \_, \_, ((ae_1\ ae_2)y = ae_3 : v)\#id$$

Then $e = (e_1\ e_2)$, $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1$, $\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae_2$, $\sigma_v'', \sigma_{id}'' \vdash e_3 \Rightarrow_s \_, \_, ae_3$, and $\mathcal{V}(ae_1) = \langle \lambda.x\ e_3, \sigma_v'', \sigma_{id}'' \rangle$.

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash ((ae_1\ ae_2)y = ae_3 : v)\#id \Rightarrow_{ex} ae_s^5, t_s^1; \textsf{assume } x = ae_s^1; t_s^3; \textsf{assume } y = ae_s^3; t_s^5$$

Then $ae_s = ae_s^5$, $t_s = t_s^1; \textsf{assume } x = ae_s^1; t_s^3; \textsf{assume } y = ae_s^3; t_s^5$, $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$, $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^3, t_s^3$, and $\mathcal{S} \vdash ae_3 \Rightarrow_{ex} ae_s^5, t_s^5$.

By assumption

$$t_s; \textsf{assume } z = ae_s \Rightarrow_r p_s$$

By definition of $\Rightarrow_r$

$$t_s^1; \textsf{assume } x = ae_s^1; t_s^3; \textsf{assume } y = ae_s^3; t_s^5 \textsf{assume } z = ae_s^5 \Rightarrow_r$$
$$p_s^1; \textsf{assume } x = e_s^1; p_s^2; \textsf{assume } y = e_s^2; p_s^3; \textsf{assume } z = e_s^3$$

Then $p_s = p_s^1; \textsf{assume } x = e_s^1; p_s^2; \textsf{assume } y = e_s^2; p_s^3; \textsf{assume } z = e_s^3$, $ae_s^1 \Rightarrow_r e_s^1$, $t_s^1 \Rightarrow_r p_s^1$, $ae_s^3 \Rightarrow_r e_s^2$, $t_s^3 \Rightarrow_r p_s^2$, $ae_s^5 \Rightarrow_r e_s^3$, and $t_s^5 \Rightarrow_r p_s^3$.

By assumption

$$\sigma'_v, \sigma'_{id} \vdash p_s \Rightarrow_s t'_s; \textsf{assume } z = ae'_s$$

By definition of $\Rightarrow_s$,

$$\sigma'_v, \sigma'_{id} \vdash p^1_s; \textsf{assume } x = e^1_s; p^2_s; \textsf{assume } y = e^2_s; p^3_s; \textsf{assume } z = e^3_s \Rightarrow_s$$
$$t^2_s; \textsf{assume } x = ae^2_s; t^4_s; \textsf{assume } y = ae^4_s; t^6_s \textsf{assume } z = ae^6_s$$

Then $t'_s = t^2_s; \textsf{assume } x = ae^2_s; t^4_s; \textsf{assume } y = ae^4_s; t^6_s$, $ae'_s = ae^6_s$, $\sigma'_v, \sigma'_{id} \vdash p^1_s; \textsf{assume } z = e^1_s \Rightarrow_s t^2_s; \textsf{assume } z = ae^2_s$, $\sigma'_v, \sigma'_{id} \vdash p^2_s; \textsf{assume } z = e^2_s \Rightarrow_s t^4_s; \textsf{assume } z = ae^4_s$, $\sigma'''_v, \sigma'''_{id} \vdash p^3_s; \textsf{assume } z = e^3_s \Rightarrow_s t^6_s; \textsf{assume } z = ae^6_s$.

By induction hypothesis

$$\exists\ e_1, p^1_s, e^1_s . \sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1 \wedge \mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae^1_s, t^1_s$$

$$\wedge t^1_s; \textsf{assume } z = ae^1_s \Rightarrow_r p^1_s; \textsf{assume } z = e^1_s$$

$$\wedge \sigma'_v, \sigma'_{id} \vdash p^1_s; \textsf{assume } z = e^1_s \Rightarrow_s t^2_s; \textsf{assume } z = ae^2_s$$

$$\Longrightarrow\ \exists\ ae'_1 . \mathcal{S} \vdash ae_1, ae^2_s, t^2_s \Rightarrow_{st} ae'_1 \wedge \sigma'_v, \sigma'_{id} \vdash e_1 \Rightarrow_s \_, \_, ae'_1$$

Because $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1$, $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae^1_s, t^1_s$, $t^1_s; \textsf{assume } z = ae^1_s \Rightarrow_r p^1_s; \textsf{assume } z = e^1_s$, and $\sigma'_v, \sigma'_{id} \vdash p^1_s; \textsf{assume } z = e^1_s \Rightarrow_s t^2_s; \textsf{assume } z = ae^2_s$.

$$\mathcal{S} \vdash ae_1, ae^2_s, t^2_s \Rightarrow_{st} ae'_1 \wedge \sigma'_v, \sigma'_{id} \vdash e_1 \Rightarrow_s \_, \_, ae'_1$$

By induction hypothesis

$$\exists\ e_2, p^2_s, e^2_s . \sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae_2 \wedge \mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae^3_s, t^3_s$$

$$\wedge t^3_s; \textsf{assume } z = ae^3_s \Rightarrow_r p^2_s; \textsf{assume } z = e^2_s$$

$$\wedge \sigma'_v, \sigma'_{id} \vdash p^2_s; \textsf{assume } z = e^2_s \Rightarrow_s t^4_s; \textsf{assume } z = ae^4_s$$

$$\Longrightarrow\ \exists\ ae'_2 . \mathcal{S} \vdash ae_2, ae^4_s, t^4_s \Rightarrow_{st} ae'_2 \wedge \sigma'_v, \sigma'_{id} \vdash e_2 \Rightarrow_s \_, \_, ae'_2$$

Because $\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae_2$, $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae^3_s, t^3_s$, $t^3_s;\textsf{assume}\ z = ae^3_s \Rightarrow_r p^2_s;\textsf{assume}\ z = e^2_s$, and $\sigma'_v, \sigma'_{id} \vdash p^2_s;\textsf{assume}\ z = e^2_s \Rightarrow_s t^4_s;\textsf{assume}\ z = ae^4_s$.

$$\mathcal{S} \vdash ae_2, ae^4_s, t^4_s \Rightarrow_{st} ae'_2 \wedge \sigma'_v, \sigma'_{id} \vdash e_2 \Rightarrow_s \_, \_, ae'_2$$

By induction hypothesis

$$\exists\ e_3, p^3_s, e^3_s.\sigma''_v, \sigma''_{id} \vdash e_3 \Rightarrow_s \_, \_, ae_3 \wedge \mathcal{S} \vdash ae_3 \Rightarrow_{ex} ae^5_s, t^5_s$$

$$\wedge t^5_s;\textsf{assume}\ z = ae^5_s \Rightarrow_r p^3_s;\textsf{assume}\ z = e^3_s$$

$$\wedge \sigma'''_v, \sigma'''_{id} \vdash p^3_s;\textsf{assume}\ z = e^3_s \Rightarrow_s t^6_s;\textsf{assume}\ z = ae^6_s$$

$$\Longrightarrow\ \exists\ ae'_3.\mathcal{S} \vdash ae_3, ae^6_s, t^6_s \Rightarrow_{st} ae'_3 \wedge \sigma'_v, \sigma'_{id} \vdash e_3 \Rightarrow_s \_, \_, ae'_3$$

Because $\sigma''_v, \sigma''_{id} \vdash e_3 \Rightarrow_s \_, \_, ae_3$, $\mathcal{S} \vdash ae_3 \Rightarrow_{ex} ae^5_s, t^5_s$, $t^5_s;\textsf{assume}\ z = ae^5_s \Rightarrow_r p^3_s;\textsf{assume}\ z = e^3_s$, and $\sigma'''_v, \sigma'''_{id} \vdash p^3_s;\textsf{assume}\ z = e^3_s \Rightarrow_s t^6_s;\textsf{assume}\ z = ae^6_s$.

$$\mathcal{S} \vdash ae_3, ae^6_s, t^6_s \Rightarrow_{st} ae'_3 \wedge \sigma'''_v, \sigma'''_{id} \vdash e_3 \Rightarrow_s \_, \_, ae'_3$$

Consider $ae' = ((ae'_1\ ae'_2)y = ae'_3 : v')\#id'$.

By definition of $\Rightarrow_{st}$, $\mathcal{S} \vdash ae_1, ae^2_s, t^2_s \Rightarrow_{st} ae'_1$, $\mathcal{S} \vdash ae_2, ae^4_s, t^4_s \Rightarrow_{st} ae'_2$, and $\mathcal{S} \vdash ae_3, ae^6_s, t^6_s \Rightarrow_{st} ae'_3$

$$\begin{gathered}\mathcal{S} \vdash ((ae_1\ ae_2)y = ae_3 : v)\#id, ((ae^2_s\ ae^4_s)y = ae^6_s : v')\#id', t'_s \\ \Rightarrow_{st} ((ae'_1\ ae'_2)y = ae'_3 : v')\#id'\end{gathered}$$

Therefore

$$\mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae'$$

By definition of $\Rightarrow_s$, $\sigma'_v, \sigma'_{id} \vdash e_1 \Rightarrow_s \_, \_, ae'_1$, $\sigma'_v, \sigma'_{id} \vdash e_2 \Rightarrow_s, \_, \_, ae'_2$, and because $\mathcal{V}(ae'_1) = \mathcal{V}(ae^2_s)$, $\mathcal{V}(ae'_1) = \mathcal{V}(ae^2_s)$ (Observation 2), $\sigma'''_v, \sigma'''_{id} \vdash e_3 \Rightarrow_s \_, \_, ae'_3$

$$\sigma'_v, \sigma'_{id} \vdash e \Rightarrow_s \_, \_, ((ae'_1\ ae'_2)y = ae'_3 : v')\#id'$$

Therefore

$$\sigma'_v, \sigma'_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

Therefore when $ae = ((ae_1\ ae_2)y = ae_3 : v)\#id$, $ID(ae_1) \notin \mathcal{S}$

$$\exists\ e, p_s.\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae \wedge \mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

$$\wedge t_s; \textsf{assume}\ z = ae_s \Rightarrow_r p_s \wedge \sigma'_v, \sigma'_{id} \vdash p_s \Rightarrow_s t'_s; \textsf{assume}\ z = ae'_s$$

$$\Longrightarrow\ \exists\ ae'.\mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae', \_ \wedge \sigma'_v, \sigma'_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

Because we have covered all cases, using induction, the lemma is true. □

Lemma 4. *Given environements $\sigma_v, \sigma_{id}, \sigma'_v$ and σ'_{id} such that* dom σ_v = dom σ_{id} = dom σ'_v = dom σ'_{id} *a partial trace t within a trace t_p(t is a suffix of trace t_p), and a valid subproblem $\mathcal{S}$ over trace t_p*

$$\exists.p, p_s\ \sigma_v, \sigma_{id} \vdash p \Rightarrow_s t \wedge \mathcal{S} \vdash t \Rightarrow_{ex} t_s \wedge p_s \Rightarrow_r t_s \wedge \sigma'_v, \sigma'_{id} \vdash p_s \Rightarrow_s t'_s$$

$$\Longrightarrow\ \exists\ t'.\mathcal{S} \vdash t, t'_s \Rightarrow_{st} t' \wedge \sigma'_v, \sigma'_{id} \vdash p \Rightarrow_s t'$$

Proof. Proof by induction

Base Case: $t = \emptyset$

By assumption

$$\sigma_v, \sigma_{id} \vdash p \Rightarrow_s t$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash \emptyset \Rightarrow_s \emptyset$$

Then $p = \emptyset$.

By assumption

$$\mathcal{S} \vdash t \Rightarrow_{ex} t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash \emptyset \Rightarrow_{ex} \emptyset$$

Then $t_s = \emptyset$.

By assumption

$$t_s \Rightarrow_r p_s$$

By definition of $\Rightarrow_r$

$$\emptyset \Rightarrow_r \emptyset$$

Then $p_s = \emptyset$.

By assumption

$$\sigma'_v, \sigma'_{id} \vdash p_s \Rightarrow_s t'_s$$

By definition of $\Rightarrow_s$

$$\sigma'_v, \sigma'_{id} \vdash \emptyset \Rightarrow_s \emptyset$$

Then $t'_s = \emptyset$.

Consider $t' = \emptyset$.

By definition of $\Rightarrow_{st}$

$$\mathcal{S} \vdash \emptyset, \emptyset \Rightarrow_{st} \emptyset$$

Therefore

$$\mathcal{S} \vdash t, t'_s \Rightarrow_{st} t'$$

By definition of $\Rightarrow_s$

$$\sigma'_v, \sigma'_{id} \vdash \emptyset \Rightarrow_s \emptyset$$

Therefore

$$\sigma'_v, \sigma'_{id} \vdash p \Rightarrow_s t'$$

Therefore when $t = \emptyset$

$$\exists . p, p_s\ \sigma_v, \sigma_{id} \vdash p \Rightarrow_s t \wedge \mathcal{S} \vdash t \Rightarrow_{ex} t_s \wedge p_s \Rightarrow_r t_s \wedge \sigma'_v, \sigma'_{id} \vdash p_s \Rightarrow_s t'_s$$

$$\implies \mathcal{S} \vdash t, t'_s \Rightarrow_{st} t' \wedge \sigma'_v, \sigma'_{id} \vdash p \Rightarrow_s t'$$

Induction Case:

Case 1: $t = \textsf{assume } x = ae; t_1$

By assumption

$$\sigma_v, \sigma_{id} \vdash p \Rightarrow_s t$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash \textsf{assume } x = e; p_1 \Rightarrow_s \textsf{assume } x = ae; t_1$$

Then $p = \textsf{assume } x = e; p_1$, $\sigma_v, \sigma_{id} \vdash e \Rightarrow_s v, id, ae$, $\sigma''_v = \sigma_v[x \to v]$, $\sigma''_{id} = \sigma_{id}[x \to id]$, and $\sigma''_v, \sigma''_{id} \vdash p_1 \Rightarrow_s t_1$.

By assumption

$$\mathcal{S} \vdash t \Rightarrow_{ex} t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash \textsf{assume } x = ae; t_1 \Rightarrow_{ex} t^1_s; \textsf{assume } x = ae_s; t^3_s$$

Then $t_s = t^1_s; \textsf{assume } x = ae_s; t^3_s$, $\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t^1_s$, and $\mathcal{S} \vdash t_1 \Rightarrow_{ex} t^3_s$.

By assumption

$$t_s \Rightarrow_r p_s$$

By definition of $\Rightarrow_r$

$$t^1_s; \textsf{assume } x = ae_s; t^3_s \Rightarrow_r p^1_s; \textsf{assume } x = e_s; p^2_s$$

Then $p_s = p^1_s; \textsf{assume } x = e_s; p^2_s$, $t^1_s \Rightarrow_r p^1_s$, $ae_s \Rightarrow_r e_s$, and $t^3_s \Rightarrow_r p^2_s$.

By assumption

$$\sigma'_v, \sigma'_{id} \vdash p_s \Rightarrow_s t'_s$$

By definition of $\Rightarrow_s$

$$\sigma'_v, \sigma'_{id} \vdash p^1_s; \textsf{assume } x = e_s; p^2_s \Rightarrow_s t^2_s; \textsf{assume } x = ae'_s; t^4_s$$

Then $t'_s = t^2_s; \textsf{assume } x = ae'_s; t^4_s$, $\sigma'_v, \sigma'_{id} \vdash p^1_s; \textsf{assume } x = e_s \Rightarrow_s t^2_s; \textsf{assume } x = ae'_s$, $\sigma'''_v = \sigma'_v[x \to \mathcal{V}(ae'_s)]$, $\sigma'''_{id} = \sigma'_{id}[x \to ID(ae'_s)]$, and $\sigma'''_v, \sigma'''_{id} \vdash p^2_s \Rightarrow_s t^4_s$ (Observation 1, variable names in t^2_s do not collide with variable names in t^4_s).

By induction hypothesis

$$\sigma''_v, \sigma''_{id} \vdash p_1 \Rightarrow_s t_1 \wedge \mathcal{S} \vdash t_1 \Rightarrow_{ex} t^3_s \wedge t^3_s \Rightarrow_r p^2_s \wedge \sigma'''_v, \sigma'''_{id} \vdash p^2_s \Rightarrow_s t^4_s$$

$$\implies \mathcal{S} \vdash t_1, t^4_s \Rightarrow_{st} t'_1 \wedge \sigma'_v, \sigma'_{id} \vdash p_1 \Rightarrow_s t'_1$$

Because $\sigma''_v, \sigma''_{id} \vdash p_1 \Rightarrow_s t_1$, $\mathcal{S} \vdash t_1 \Rightarrow_{ex} t^3_s$, $\mathcal{S} \vdash t_1 \Rightarrow_{ex} t^3_s$, $t^3_s \Rightarrow_r p^2_s$, and $\sigma'''_v, \sigma'''_{id} \vdash p^2_s \Rightarrow_s t^4_s$,

$$\mathcal{S} \vdash t_1, t^4_s \Rightarrow_{st} t'_1 \wedge \sigma'_v, \sigma'_{id} \vdash p_1 \Rightarrow_s t'_1$$

By statement 3

$$\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae \wedge \mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t^1_s$$

$$\wedge t^1_s; \textsf{assume } z = ae_s \Rightarrow_r p^1_s; \textsf{assume } z = e_s$$

$$\wedge \sigma'_v, \sigma'_{id} \vdash p^1_s; \textsf{assume } z = e_s \Rightarrow_s t^2_s; \textsf{assume } z = ae'_s$$

$$\implies \mathcal{S} \vdash ae, ae'_s, t^2_s \Rightarrow_{st} ae' \wedge \sigma'_v, \sigma'_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

Because $\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae$, $\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t^1_s$, $t^1_s; \textsf{assume } z = ae_s \Rightarrow_r p^1_s; \textsf{assume } z = e_s$ and $\sigma'_v, \sigma'_{id} \vdash p^1_s; \textsf{assume } z = e_s \Rightarrow_s t^2_s; \textsf{assume } z = ae'_s$,

$$\mathcal{S} \vdash ae, ae'_s, t^2_s \Rightarrow_{st} ae' \wedge \sigma'_v, \sigma'_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

Consider $t' = \textsf{assume } x = ae'; t'_1$.

By definition of $\Rightarrow_{st}$, $\mathcal{S} \vdash t_1, t_s^4 \Rightarrow_{st} t_1'$, and $\mathcal{S} \vdash ae, ae_s', t_s^2 \Rightarrow_{st} ae'$,

$$\mathcal{S} \vdash \textsf{assume } x = ae, t_s^2; \textsf{assume } x = ae_s'; t_s^4 \Rightarrow_{st} \textsf{assume } x = ae'; t_1'$$

Therefore

$$\mathcal{S} \vdash t, t_s' \Rightarrow_{st} t'$$

By definition of $\Rightarrow_s$, $\sigma_v', \sigma_{id}' \vdash e \Rightarrow_s \_, \_, ae'$, and $\sigma_v''', \sigma_{id}''' \vdash p_s^2 \Rightarrow_s t_s^4$ ($\mathcal{V}(ae') = \mathcal{V}(ae_s')$ and $ID(ae') = ID(ae_s')$ Observation 2)

$$\sigma_v', \sigma_{id}' \vdash \textsf{assume } x = e; p_1 \Rightarrow_s \textsf{assume } x = ae'; t_1'$$

Therefore

$$\sigma_v', \sigma_{id}' \vdash p \Rightarrow_s t'$$

Therefore when $t = \textsf{assume } x = ae; t_1$

$$\exists . p, p_s\ \sigma_v, \sigma_{id} \vdash p \Rightarrow_s t \wedge \mathcal{S} \vdash t \Rightarrow_{ex} t_s \wedge p_s \Rightarrow_r t_s \wedge \sigma_v', \sigma_{id}' \vdash p_s \Rightarrow_s t_s'$$

$$\implies \mathcal{S} \vdash t, t_s' \Rightarrow_{st} t' \wedge \sigma_v', \sigma_{id}' \vdash p \Rightarrow_s t'$$

Case 2: $t = \textsf{observe}(\textsf{Dist}(ae) = e_v); t_1$

By assumption

$$\sigma_v, \sigma_{id} \vdash p \Rightarrow_s t$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash \textsf{observe}(\textsf{Dist}(e) = e_v); p_1 \Rightarrow_s \textsf{observe}(\textsf{Dist}(ae) = e_v); t_1$$

Then $p = \textsf{observe}(\textsf{Dist}(e) = e_v); p_1$, $\sigma_v, \sigma_{id} \vdash e \Rightarrow_s v, id, ae$, and $\sigma_v, \sigma_{id} \vdash p_1 \Rightarrow_s t_1$.

By assumption

$$\mathcal{S} \vdash t \Rightarrow_{ex} t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash \mathsf{observe}(\mathsf{Dist}(ae) = e_v); t_1 \Rightarrow_{ex} t_s^1; \mathsf{observe}(\mathsf{Dist}(ae_s) = e_v); t_s^3$$

Then $t_s = t_s^1; \mathsf{observe}(\mathsf{Dist}(ae_s) = e_v); t_s^3$, $\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s^1$, and $\mathcal{S} \vdash t_1 \Rightarrow_{ex} t_s^3$.

By assumption

$$t_s \Rightarrow_r p_s$$

By definition of $\Rightarrow_r$

$$t_s^1; \mathsf{observe}(\mathsf{Dist}(ae_s) = e_v); t_s^3 \Rightarrow_r p_s^1; \mathsf{observe}(\mathsf{Dist}(e_s) = e_v); p_s^2$$

Then $p_s = p_s^1; \mathsf{observe}(\mathsf{Dist}(ae_s) = e_v); p_s^2$, $t_s^1 \Rightarrow_r p_s^1$, $ae_s \Rightarrow_r e_s$, and $t_s^3 \Rightarrow_r p_s^2$.

By assumption

$$\sigma_v', \sigma_{id}' \vdash p_s \Rightarrow_s t_s'$$

By definition of $\Rightarrow_s$

$$\sigma_v', \sigma_{id}' \vdash p_s^1; \mathsf{observe}(\mathsf{Dist}(e_s) = e_v); p_s^2 \Rightarrow_s t_s^2; \mathsf{observe}(\mathsf{Dist}(ae_s') = e_v); t_s^4$$

Then $t_s' = t_s^2; \mathsf{observe}(\mathsf{Dist}(ae_s') = e_v); t_s^4$, $\sigma_v', \sigma_{id}' \vdash p_s^1; \mathsf{observe}(\mathsf{Dist}(e_s) = e_v) \Rightarrow_s t_s^2; \mathsf{observe}(\mathsf{Dist}(ae_s') = e_v)$, and $\sigma_v', \sigma_{id}' \vdash p_s^2 \Rightarrow_s t_s^4$ (Observation 1, variable names in t_s^2 do not collide with variable names in t_s^4).

By induction hypothesis

$$\sigma_v, \sigma_{id} \vdash p_1 \Rightarrow_s t_1 \wedge \mathcal{S} \vdash t_1 \Rightarrow_{ex} t_s^3 \wedge t_s^3 \Rightarrow_r p_s^2 \wedge \sigma_v', \sigma_{id}' \vdash p_s^2 \Rightarrow_s t_s^4$$

$$\Longrightarrow \mathcal{S} \vdash t_1, t_s^4 \Rightarrow_{st} t_1' \wedge \sigma_v', \sigma_{id}' \vdash p_1 \Rightarrow_s t_1'$$

Because $\sigma_v, \sigma_{id} \vdash p_1 \Rightarrow_s t_1$, $\mathcal{S} \vdash t_1 \Rightarrow_{ex} t_s^3$, $\mathcal{S} \vdash t_1 \Rightarrow_{ex} t_s^3$, $t_s^3 \Rightarrow_r p_s^2$, and $\sigma_v', \sigma_{id}' \vdash p_s^2 \Rightarrow_s t_s^4$,

$$\mathcal{S} \vdash t_1, t_s^4 \Rightarrow_{st} t_1' \wedge \sigma_v', \sigma_{id}' \vdash p_1 \Rightarrow_s t_1'$$

By statement 3

$$\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae \wedge \mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s^1$$

$$\wedge t_s^1; \textsf{observe}(\textsf{Dist}(ae_s) = e_v) \Rightarrow_r p_s^1; \textsf{observe}(\textsf{Dist}(e_s) = e_v)$$

$$\wedge \sigma_v', \sigma_{id}' \vdash p_s^1; \textsf{observe}(\textsf{Dist}(e_s) = e_v) \Rightarrow_s t_s^2; \textsf{observe}(\textsf{Dist}(e_s) = ae_v')$$

$$\implies \mathcal{S} \vdash ae, ae_s', t_s^2 \Rightarrow_{st} ae' \wedge \sigma_v', \sigma_{id}' \vdash e \Rightarrow_s \_, \_, ae'$$

Because $\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae$, $\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s^1$, $t_s^1; \textsf{observe}(\textsf{Dist}(ae_s) = e_v) \Rightarrow_r p_s^1; \textsf{observe}(\textsf{Dist}(e_s) = e_v)$ and $\sigma_v', \sigma_{id}' \vdash p_s^1; \textsf{observe}(\textsf{Dist}(e_s) = e_v) \Rightarrow_s t_s^2; \textsf{observe}(\textsf{Dist}(ae_s') = e_v)$,

$$\mathcal{S} \vdash ae, ae_s', t_s^2 \Rightarrow_{st} ae' \wedge \sigma_v', \sigma_{id}' \vdash e \Rightarrow_s \_, \_, ae'$$

Consider $t' = \textsf{observe}(\textsf{Dist}(ae') = e_v); t_1'$.

By definition of $\Rightarrow_{st}$, $\mathcal{S} \vdash t_1, t_s^4 \Rightarrow_{st} t_1'$, and $\mathcal{S} \vdash ae, ae_s', t_s^2 \Rightarrow_{st} ae'$,

$$\mathcal{S} \vdash \textsf{observe}(\textsf{Dist}(ae) = e_v), t_s^2; \textsf{observe}(\textsf{Dist}(ae_s') = e_v); t_s^4 \Rightarrow_{st} \textsf{observe}(\textsf{Dist}(ae') = e_v); t_1'$$

Therefore

$$\mathcal{S} \vdash t, t_s' \Rightarrow_{st} t'$$

By definition of $\Rightarrow_s$, $\sigma_v', \sigma_{id}' \vdash e \Rightarrow_s \_, \_, ae'$, and $\sigma_v', \sigma_{id}' \vdash p_s^2 \Rightarrow_s t_s^4$

$$\sigma_v', \sigma_{id}' \vdash \textsf{observe}(\textsf{Dist}(e) = e_v); p_1 \Rightarrow_s \textsf{observe}(\textsf{Dist}(ae') = e_v); t_1'$$

Therefore

$$\sigma_v', \sigma_{id}' \vdash p \Rightarrow_s t'$$

Therefore when $t = \textsf{observe}(\textsf{Dist}(ae) = e_v); t_1$

$$\exists . p, p_s\ \sigma_v, \sigma_{id} \vdash p \Rightarrow_s t \wedge \mathcal{S} \vdash t \Rightarrow_{ex} t_s \wedge p_s \Rightarrow_r t_s \wedge \sigma_v', \sigma_{id}' \vdash p_s \Rightarrow_s t_s'$$

$$\implies \mathcal{S} \vdash t, t_s' \Rightarrow_{st} t' \wedge \sigma_v', \sigma_{id}' \vdash p \Rightarrow_s t'$$

Because we have covered all cases, using induction, the below statement is true.

$$\exists . p, p_s \; \sigma_v, \sigma_{id} \vdash p \Rightarrow_s t \wedge \mathcal{S} \vdash t \Rightarrow_{ex} t_s \wedge p_s \Rightarrow_r t_s \wedge \sigma_v', \sigma_{id}' \vdash p_s \Rightarrow_s t_s'$$

$$\implies \mathcal{S} \vdash t, t_s' \Rightarrow_{st} t' \wedge \sigma_v', \sigma_{id}' \vdash p \Rightarrow_s t'$$

□

Lemma 5. *Given a valid trace t and a valid subproblem $\mathcal{S}$ and subtrace $t_s = \mathsf{ExtractTrace}(t, \mathcal{S})$ for all possible subtraces t_s' :*

$t_s' \in \mathsf{Traces}(\mathsf{Program}(t_s)) \implies \exists\, t'.t' = \mathsf{StitchTrace}(t, t_s', \mathcal{S}) \;\wedge\; t' \in \mathsf{Traces}(\mathsf{Program}(t))$

Proof. Since statement 4 is true for all $\sigma_v, \sigma_{id}, \sigma_v'$, and σ_{id}', given $\mathsf{dom}\ \sigma_v = \mathsf{dom}\ \sigma_{id} = \mathsf{dom}\ \sigma_v' = \mathsf{dom}\ \sigma_{id}'$, replacing $\sigma_v, \sigma_{id}, \sigma_v'$, and σ_{id}' with $\emptyset$ (empty environment) will result in

$\exists . p, p_s \; t \in \mathsf{Traces}(p) \wedge \mathcal{S} \vdash t \Rightarrow_{ex} t_s \wedge t_s, t_s' \in \mathsf{Traces}(p_s) \implies \mathcal{S} \vdash t, t_s' \Rightarrow_{st} t' \wedge t' \in \mathsf{Traces}(p)$

□

Theorem 1. *Given a valid trace t, a valid subproblem $\mathcal{S}$, subtrace $t_s = \mathsf{ExtractTrace}(t, \mathcal{S})$. For all possible subtraces t_s', $t_s' \in \mathsf{Traces}(\mathsf{Program}(t_s))$ implies there exist a trace t' such that:*

- $t' = \mathsf{StitchTrace}(t, t_s', \mathcal{S})$
- $t' \in \mathsf{Traces}(\mathsf{Program}(t))$
- $\mathcal{S} \vdash t \equiv t'$

Proof. From Corollary 1 and Lemma 5. □

3.2 Completeness

Within this section we prove that our interface is complete, i.e. given a valid trace t, a valid subproblem $\mathcal{S}$ on t, a subtrace $t_s = \mathsf{ExtractTrace}(t, t_s, \mathcal{S})$, for any trace t' which can be achived from entangled subproblem interface (i.e. $t' \in \mathsf{Traces}(\mathsf{Program}(t))$ and $\mathcal{S} \vdash t \equiv t'$), there exists a subtrace $t'_s \in \mathsf{Traces}(\mathsf{Program}(t_s))$ such that $t' = \mathsf{StitchTrace}(t, t'_s, \mathcal{S})$.

Formally, given valid trace t, a valid subproblem $\mathcal{S}$ on t, a subtrace t_s, and any trace t'

$$t_s = \mathsf{ExtractTrace}(t, \mathcal{S}) \ \wedge \ t' \in \mathsf{Traces}(\mathsf{Program}(t)) \ \wedge \ \mathcal{S} \vdash t \equiv t'$$

$$\implies \exists \, t'_s. \ t'_s \in \mathsf{Traces}(\mathsf{Program}(t_s)) \ \wedge \ t' = \mathsf{StitchTrace}(t, t'_s, \mathcal{S})$$

We need to prove a few lemmas which will aid us in proving the above statement.

Lemma 6. *Given an augmented expression ae and a subproblem $\mathcal{S}$, a subtrace t_s, subaugmented expression ae_s and an augmented expressions ae' such that*

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s \wedge \mathcal{S} \vdash ae \equiv ae' \wedge \exists \ e. \ ae \Rightarrow_r e \wedge \sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

then, there exists an ae'_s, t'_s, p_s and e_s such that

$$\mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae' \wedge t_s \Rightarrow_r p_s \ \wedge \ ae_s \Rightarrow_r e_s$$

$$\wedge \ \sigma_v, \sigma_{id} \vdash p_s; \mathsf{assume} \ z = e_s \Rightarrow_s t'_s; \mathsf{assume} \ z = ae'_s$$

Proof. Proof by Induction

Base Case:

Case 1: $ae = (x : x)\#id$

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash (x : x)\#id \Rightarrow_{ex} (x : x)\#id, \emptyset$$

Then $ae_s = (x : x)\#id$ and $t_s = \emptyset$.

By assumption

$$\mathcal{S} \vdash ae \equiv ae'$$

By definition of $\equiv$

$$\mathcal{S} \vdash (x : x)\#id \equiv (x : x)\#id'$$

Then $ae' = (x : x)\#id'$.

By assumption

$$ae \Rightarrow_r e$$

By definition of $\Rightarrow_r$

$$(x : x)\#id \Rightarrow_r x$$

Then $e = x$.

By assumption

$$\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash x \Rightarrow_s \_, \_, (x : x)\#id'$$

Then $x \notin \mathsf{dom}\ \sigma_v$ and id' is a unique id.

Consider $ae'_s = (x : x)\#id'$, $t'_s = \emptyset$, $p_s = \emptyset$ and $e_s = x$. By definition of $\Rightarrow_{st}$

$$\mathcal{S} \vdash (x : x)\#id, (x : x)\#id', \emptyset \Rightarrow_{st} (x : x)\#id'$$

Therefore

$$\mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae'$$

By definition of $\Rightarrow_r$

$$\emptyset \Rightarrow_r \emptyset$$

Therefore

$$t_s \Rightarrow_r p_s$$

By definition of $\Rightarrow_r$

$$(x : x)\#id \Rightarrow_r x$$

Therefore

$$ae_s \Rightarrow_r e_s$$

Because $x \notin \textsf{dom}\ \sigma_v$, the definition of $\Rightarrow_s$ implies

$$\sigma_v, \sigma_{id} \vdash \textsf{assume}\ z = x \Rightarrow_s \textsf{assume}\ z = (x : x)\#id'$$

Therefore

$$\sigma_v, \sigma_{id} \vdash p_s; \textsf{assume}\ z = e_s \Rightarrow_s t_s; \textsf{assume}\ z = ae_s'$$

Therefore when $ae = (x : x)\#id$,

$\forall t_s, ae_s, ae'\ \mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s\ \wedge\ \mathcal{S} \vdash ae \equiv ae'\ \exists e.\ ae \Rightarrow_r e\ \wedge\ \sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$

implies

$$\exists ae_s', t_s', p_s, e_s.\ \mathcal{S} \vdash ae, ae_s', t_s' \Rightarrow_{st} ae'\ \wedge\ t_s \Rightarrow_r p_s\ \wedge\ ae_s \Rightarrow_r e_s$$

$$\wedge\ \sigma_v, \sigma_{id} \vdash p_s; \textsf{assume}\ z = e_s \Rightarrow_s t_s'; \textsf{assume}\ z = ae_s'$$

Case 2: $ae = (x(id_v) : v)\#id$

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash (x(id_v) : v)\#id \Rightarrow_{ex} (x(id_v) : v)\#id, \emptyset$$

Then $ae_s = (x(id_v) : v)\#id$ and $t_s = \emptyset$.

By assumption

$$\mathcal{S} \vdash ae \equiv ae'$$

By definition of $\equiv$

$$\mathcal{S} \vdash (x(id_v) : v)\#id \equiv (x(id'_v) : v')\#id'$$

Then $ae' = (x(id'_v) : v)\#id'$.

By assumption

$$ae \Rightarrow_r e$$

By definition of $\Rightarrow_r$

$$(x(id_v) : v)\#id \Rightarrow_r x$$

Then $e = x$.

By assumption

$$\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash x \Rightarrow_s \_, \_, (x(id'_v) : v')\#id'$$

Then $\sigma_v(x) = v', \sigma_{id}(x) = id'_v$, and id' is a unique id.

Consider $ae'_s = (x(id'_v) : v')\#id'$, $t'_s = \emptyset$, $p_s = \emptyset$ and $e_s = x$. By definition of $\Rightarrow_{st}$

$$\mathcal{S} \vdash (x(id_v) : v)\#id, (x(id'_v) : v')\#id', \emptyset \Rightarrow_{st} (x(id'_v) : v')\#id'$$

Therefore

$$\mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae'$$

By definition of $\Rightarrow_r$

$$\emptyset \Rightarrow_r \emptyset$$

Therefore

$$t_s \Rightarrow_r p_s$$

By definition of $\Rightarrow_r$

$$(x(id_v) : v)\#id \Rightarrow_r x$$

Therefore

$$ae_s \Rightarrow_r e_s$$

Because $\sigma_v(x) = v', \sigma_{id}(x) = id'_v$, the definition of $\Rightarrow_s$ implies

$$\sigma_v, \sigma_{id} \vdash \textsf{assume } z = x \Rightarrow_s \textsf{assume } z = (x(id'_v) : v')\#id'$$

Therefore

$$\sigma_v, \sigma_{id} \vdash p_s; \textsf{assume } z = e_s \Rightarrow_s t_s; \textsf{assume } z = ae'_s$$

Therefore when $ae = (x(id_v) : v)\#id$,

$$\forall t_s, ae_s, ae' \ \mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s \ \wedge \ \mathcal{S} \vdash ae \equiv ae' \ \exists e.\ ae \Rightarrow_r e \ \wedge \ \sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

implies

$$\exists ae'_s, t'_s, p_s, e_s.\ \mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae' \ \wedge \ t_s \Rightarrow_r p_s \ \wedge \ ae_s \Rightarrow_r e_s$$

$$\wedge \ \sigma_v, \sigma_{id} \vdash p_s; \textsf{assume } z = e_s \Rightarrow_s t'_s; \textsf{assume } z = ae'_s$$

Case 3: $ae = (\lambda.x\ e' : v)\#id$

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash (\lambda.x\ e' : v)\#id \Rightarrow_{ex} (\lambda.x\ e' : v)\#id, \emptyset$$

Then $ae_s = (\lambda.x\ e' : v)\#id$ and $t_s = \emptyset$.

By assumption

$$\mathcal{S} \vdash ae \equiv ae'$$

By definition of $\equiv$

$$\mathcal{S} \vdash (\lambda.x\ e' : v)\#id \equiv (\lambda.x\ e' : v')\#id'$$

Then $ae' = (\lambda.x\ e' : v)\#id'$.

By assumption

$$ae \Rightarrow_r e$$

By definition of $\Rightarrow_r$

$$(\lambda.x\ e' : v)\#id \Rightarrow_r \lambda.x\ e'$$

Then $e = \lambda.x\ e'$.

By assumption

$$\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash \lambda.x\ e' \Rightarrow_s \_, \_, (\lambda.x\ e' : v')\#id'$$

Then id' is a unique id.

Consider $ae'_s = (\lambda.x\ e' : v')\#id'$, $t'_s = \emptyset$, $p_s = \emptyset$ and $e_s = \lambda.x\ e'$. By definition of $\Rightarrow_{st}$

$$\mathcal{S} \vdash (\lambda.x\ e' : v)\#id, (\lambda.x\ e' : v')\#id', \emptyset \Rightarrow_{st} (\lambda.x\ e' : v')\#id'$$

Therefore

$$\mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae'$$

By definition of $\Rightarrow_r$

$$\emptyset \Rightarrow_r \emptyset$$

Therefore

$$t_s \Rightarrow_r p_s$$

By definition of $\Rightarrow_r$

$$(\lambda.x\ e' : v)\#id \Rightarrow_r \lambda.x\ e'$$

Therefore

$$ae_s \Rightarrow_r e_s$$

Because $\sigma_v, \sigma_{id} \vdash \lambda.x\ e' \Rightarrow_s \_, \_, (\lambda.x\ e' : v')\#id'$, the definition of $\Rightarrow_s$ implies

$$\sigma_v, \sigma_{id} \vdash \textsf{assume } z = x \Rightarrow_s \textsf{assume } z = (x(id'_v) : v')\#id'$$

Therefore

$$\sigma_v, \sigma_{id} \vdash p_s; \textsf{assume } z = e_s \Rightarrow_s t_s; \textsf{assume } z = ae'_s$$

Therefore when $ae = (\lambda.x\ e' : v)\#id$,

$\forall t_s, ae_s, ae'\ \mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s\ \wedge\ \mathcal{S} \vdash ae \equiv ae'\ \exists e.\ ae \Rightarrow_r e\ \wedge\ \sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$

implies

$$\exists ae'_s, t'_s, p_s, e_s.\ \mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae'\ \wedge\ t_s \Rightarrow_r p_s\ \wedge\ ae_s \Rightarrow_r e_s$$

$$\wedge\ \sigma_v, \sigma_{id} \vdash p_s; \textsf{assume } z = e_s \Rightarrow_s t'_s; \textsf{assume } z = ae'_s$$

Induction Case:

Case 1: $ae = (\textsf{Dist}(ae_1\#id_e) = ae_2 : v)\#id$ and $id_e \in \mathcal{S}$

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash (\textsf{Dist}(ae_1\#id_e) = ae_2 : v)\#id \Rightarrow_{ex} (\textsf{Dist}(ae^1_s\#id_e) = ae_2 : v)\#id, t_s$$

Then $ae_s = (\textsf{Dist}(ae^1_s\#id_e) = ae_2 : v)\#id$, $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae^1_s, t^1_s$ and $t_s = t^1_s$.

By assumption

$$\mathcal{S} \vdash ae \equiv ae'$$

By definition of $\equiv$

$$\mathcal{S} \vdash (\textsf{Dist}(ae_1\#id_e) = ae_2 : v)\#id \equiv (\textsf{Dist}(ae'_1\#id'_e) = ae'_2 : v')\#id'$$

Then $ae' = (\mathsf{Dist}(ae'_1\#id'_e) = ae'_2 : v')\#id'$ and $\mathcal{S} \vdash ae_1 \equiv ae'_1$.

By assumption

$$\exists e.\ ae \Rightarrow_r e$$

By definition of $\Rightarrow_r$

$$(\mathsf{Dist}(ae_1\#id_e) = ae_2 : v)\#id \Rightarrow_r \mathsf{Dist}(e_1)$$

Then $e = \mathsf{Dist}(e_1)$ and $ae_1 \Rightarrow_r e_1$.

By assumption

$$\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash \mathsf{Dist}(e_1) \Rightarrow_s \_, \_, (\mathsf{Dist}(ae'_1\#id'_e) = ae'_2 : v')\#id'$$

Then $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae'_1$, $e_2 \in \mathsf{dom}\ \mathsf{Dist}(v')$, and $\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae'_2$.

By induction hypothesis

$$\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae^1_s, t^1_s \ \wedge\ \mathcal{S} \vdash ae_1 \equiv ae'_1 \ \wedge\ ae_1 \Rightarrow_r e_1 \ \wedge \sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae'_1$$

$$\implies \mathcal{S} \vdash ae_1, ae^2_s, t^2_s \Rightarrow_{st} ae'_1 \ \wedge\ t^1_s \Rightarrow_r p^1_s \ \wedge\ ae^1_s \Rightarrow_r e^1_s$$

$$\wedge\ \sigma_v, \sigma_{id} \vdash p^1_s; \mathsf{assume}\ z = e^1_s \Rightarrow_s t^2_s; \mathsf{assume}\ z = ae^2_s$$

Because $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae^1_s, t^1_s$, $\mathcal{S} \vdash ae_1 \equiv ae'_1$, $ae_1 \Rightarrow_r e_1$, and $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae'_1$,

$$\mathcal{S} \vdash ae_1, ae^2_s, t^2_s \Rightarrow_{st} ae'_1 \ \wedge\ t^1_s \Rightarrow_r p^1_s \ \wedge\ ae^1_s \Rightarrow_r e^1_s$$

$$\wedge\ \sigma_v, \sigma_{id} \vdash p^1_s; \mathsf{assume}\ z = e^1_s \Rightarrow_s t^2_s; \mathsf{assume}\ z = ae^2_s$$

Consider $ae'_s = (\mathsf{Dist}(ae^2_s\#id_e) = ae'_2 : v')\#id'$, $t'_s = t^2_s$, $p_s = p^1_s$, and $e_s = \mathsf{Dist}(e^1_s)$.

Because $\mathcal{S} \vdash ae_1, ae_s^2, t_s^2 \Rightarrow_{st} ae_1'$ and $id_e \in \mathcal{S}$, the definition of $\Rightarrow_{st}$ implies

$$\mathcal{S} \vdash (\mathsf{Dist}(ae_1\#id_e) = ae_2 : v)\#id, (\mathsf{Dist}(ae_s^2\#id_e') = ae_2' : v')\#id', t_s' \\ \Rightarrow_{st} (\mathsf{Dist}(ae_1'\#id_e') = ae_2' : v')\#id'$$

Therefore

$$\mathcal{S} \vdash ae, ae_s', t_s' \Rightarrow_{st} ae'$$

Because $t_s^1 \Rightarrow_r p_s^1$,

$$t_s \Rightarrow_r p_s$$

Because $ae_s^1 \Rightarrow_r e_s^1$, the definition of $\Rightarrow_r$ implies

$$(\mathsf{Dist}(ae_s^1\#id_e) = ae_2 : v)\#id \implies \mathsf{Dist}(e_s^1)$$

Therefore

$$ae_s \Rightarrow_r e_s$$

Because $\sigma_v, \sigma_{id} \vdash p_s^1; \mathsf{assume}\ z = e_s^1 \Rightarrow_s t_s^2; \mathsf{assume}\ z = ae_s^2$, $\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae_2'$, and all variable names introduced by t_s^2 do not conflict with variable names in ae_2' (Observation 1), the definition of $\Rightarrow_s$ implies

$$\sigma_v, \sigma_{id} \vdash p_s; \mathsf{assume}\ z = \mathsf{Dist}(e_s^1) \Rightarrow_s t_s'; \mathsf{assume}\ z = (\mathsf{Dist}(ae_s^1\#id_e') = ae_2' : v')\#id'$$

Therefore

$$\sigma_v, \sigma_{id} \vdash p_s; \mathsf{assume}\ z = e_s \Rightarrow_s t_s'; \mathsf{assume}\ z = ae_s'$$

Therefore when $ae = (\mathsf{Dist}(ae_1\#id_e) = ae_2 : v)\#id$ and $id_e \in \mathcal{S}$, and assuming induction hypothesis,

$$\forall t_s, ae_s, ae'\ \mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s \ \wedge\ \mathcal{S} \vdash ae \equiv ae'\ \exists e.\ ae \Rightarrow_r e \ \wedge\ \sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

$$\implies \exists ae_s', t_s', p_s, e_s.\ \mathcal{S} \vdash ae, ae_s', t_s' \Rightarrow_{st} ae' \ \wedge\ t_s \Rightarrow_r p_s \ \wedge\ ae_s \Rightarrow_r e_s$$

$$\wedge\ \sigma_v, \sigma_{id} \vdash p_s; \mathsf{assume}\ z = e_s \Rightarrow_s t'_s; \mathsf{assume}\ z = ae'_s$$

Case 2: $ae = (\mathsf{Dist}(ae_1\#id_e) = ae_2 : v)\#id$ and $id_e \notin \mathcal{S}$

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash (\mathsf{Dist}(ae_1\#id_e) = ae_2 : v)\#id \Rightarrow_{ex} (ae^3_s : v)\#id, t_s$$

Then $ae_s = (ae^3_s : v)\#id$, $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae^1_s, t^1_s$, $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae^3_s, t^3_s$, $ae_2 \Rightarrow_r e_2$, and $t_s = t^1_s; \mathsf{observe}(\mathsf{Dist}(ae^1_s) = e_2); t^3_s$.

By assumption

$$\mathcal{S} \vdash ae \equiv ae'$$

By definition of $\equiv$

$$\mathcal{S} \vdash (\mathsf{Dist}(ae_1\#id_e) = ae_2 : v)\#id \equiv (\mathsf{Dist}(ae'_1\#id_e) = ae'_2 : v')\#id'$$

Then $ae' = (\mathsf{Dist}(ae'_1\#id_e) = ae'_2 : v')\#id'$, $\mathcal{S} \vdash ae_1 \equiv ae'_1$, $\mathcal{S} \vdash ae_2 \equiv ae'_2$.

By assumption

$$\exists e.\ ae \Rightarrow_r e$$

By definition of $\Rightarrow_r$

$$(\mathsf{Dist}(ae_1\#id_e) = ae_2 : v)\#id \Rightarrow_r \mathsf{Dist}(e_1)$$

Then $e = \mathsf{Dist}(e_1)$ and $ae_1 \Rightarrow_r e_1$.

By assumption

$$\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash \mathsf{Dist}(e_1) \Rightarrow_s \_, \_, (\mathsf{Dist}(ae'_1\#id_e) = ae'_2 : v')\#id'$$

Then $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae'_1$, $e_2 \in \textsf{dom Dist}(v')$, and $\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae'_2$.

By induction hypothesis

$$\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae^1_s, t^1_s \ \wedge \ \mathcal{S} \vdash ae_1 \equiv ae'_1 \ \wedge \ ae_1 \Rightarrow_r e_1 \ \wedge \sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae'_1$$

$$\implies \mathcal{S} \vdash ae_1, ae^2_s, t^2_s \Rightarrow_{st} ae'_1 \ \wedge \ t^1_s \Rightarrow_r p^1_s \ \wedge \ ae^1_s \Rightarrow_r e^1_s$$

$$\wedge \ \sigma_v, \sigma_{id} \vdash p^1_s; \textsf{assume}\ z = e^1_s \Rightarrow_s t^2_s; \textsf{assume}\ z = ae^2_s$$

Because $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae^1_s, t^1_s$, $\mathcal{S} \vdash ae_1 \equiv ae'_1$, $ae_1 \Rightarrow_r e_1$, and $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae'_1$,

$$\mathcal{S} \vdash ae_1, ae^2_s, t^2_s \Rightarrow_{st} ae'_1 \ \wedge \ t^1_s \Rightarrow_r p^1_s \ \wedge \ ae^1_s \Rightarrow_r e^1_s$$

$$\wedge \ \sigma_v, \sigma_{id} \vdash p^1_s; \textsf{assume}\ z = e^1_s \Rightarrow_s t^2_s; \textsf{assume}\ z = ae^2_s$$

By induction hypothesis

$$\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae^3_s, t^3_s \ \wedge \ \mathcal{S} \vdash ae_2 \equiv ae'_2 \ \wedge \ ae_2 \Rightarrow_r e_2 \ \wedge \sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae'_2$$

$$\implies \mathcal{S} \vdash ae_2, ae^4_s, t^4_s \Rightarrow_{st} ae'_2 \ \wedge \ t^3_s \Rightarrow_r p^2_s \ \wedge \ ae^3_s \Rightarrow_r e^2_s$$

$$\wedge \ \sigma_v, \sigma_{id} \vdash p^2_s; \textsf{assume}\ z = e^2_s \Rightarrow_s t^4_s; \textsf{assume}\ z = ae^4_s$$

Because $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae^3_s, t^3_s$, $\mathcal{S} \vdash ae_2 \equiv ae'_2$, $ae_2 \Rightarrow_r e_2$, and $\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae'_2$,

$$\mathcal{S} \vdash ae_2, ae^4_s, t^4_s \Rightarrow_{st} ae'_2 \ \wedge \ t^3_s \Rightarrow_r p^2_s \ \wedge \ ae^3_s \Rightarrow_r e^2_s$$

$$\wedge \ \sigma_v, \sigma_{id} \vdash p^2_s; \textsf{assume}\ z = e^2_s \Rightarrow_s t^4_s; \textsf{assume}\ z = ae^4_s$$

Consider $ae'_s = (ae^4_s : v')\#id'$, $t'_s = t^2_s; \textsf{observe}(\textsf{Dist}(ae^2_s) = e_2); t^4_s$, $p_s = p^1_s; \textsf{observe}(\textsf{Dist}(e^1_s) = e_2); p^2_s$, and $e_s = e^2_s$. Because $\mathcal{S} \vdash ae_1, ae^2_s, t^2_s \Rightarrow_{st} ae'_1$, $\mathcal{S} \vdash ae_2, ae^4_s, t^4_s \Rightarrow_{st} ae'_s$, and $id_e \in \mathcal{S}$, the definition of $\Rightarrow_{st}$ implies

$$\mathcal{S} \vdash (\textsf{Dist}(ae_1\#id_e) = ae_2 : v)\#id, (ae^4_s : v')\#id', t'_s \Rightarrow_{st} (\textsf{Dist}(ae'_1\#id_e) = ae'_2 : v')\#id'$$

Therefore

$$\mathcal{S} \vdash ae, ae'_s, t'_s \Rrightarrow_{st} ae'$$

Because $t^1_s \Rrightarrow_r p^1_s$, $t^3_s \Rrightarrow_r p^2_s$, $ae^1_s \Rrightarrow_r e^1_s$, definition of $\Rrightarrow_r$ implies

$$t^1_s; \mathsf{observe}(\mathsf{Dist}(ae^1_s) = e_2); t^3_s \Rrightarrow_r p^1_s; \mathsf{observe}(\mathsf{Dist}(e^1_s) = e_2); p^2_s$$

Therefore

$$t_s \Rrightarrow_r p_s$$

Because $ae^3_s \Rrightarrow_r e^2_s$, the definition of $\Rrightarrow_r$ implies

$$(ae^3_s : v)\#id \implies e^2_s$$

Therefore

$$ae_s \Rrightarrow_r e_s$$

Because $\sigma_v, \sigma_{id} \vdash p^1_s; \mathsf{assume}\ z = e^1_s \Rrightarrow_s t^2_s; \mathsf{assume}\ z = ae^2_s$, $\sigma_v, \sigma_{id} \vdash p^2_s; \mathsf{assume}\ z = e^2_s \Rrightarrow_s t^4_s; \mathsf{assume}\ z = ae^4_s$, and all variable names introduced by t^2_s do not conflict with variable names in ae^4_s and t^4_s (Observation 1), the definition of $\Rrightarrow_s$ implies

$$\sigma_v, \sigma_{id} \vdash p_s; \mathsf{assume}\ z = e^2_s \Rrightarrow_s t'_s; \mathsf{assume}\ z = ae^4_s$$

Therefore

$$\sigma_v, \sigma_{id} \vdash p_s; \mathsf{assume}\ z = e_s \Rrightarrow_s t'_s; \mathsf{assume}\ z = ae'_s$$

Therefore when $ae = (\mathsf{Dist}(ae_1\$id_e) = ae_2 : v)\#id$ and $id_e \in \mathcal{S}$, and assuming induction hypothesis,

$$\forall t_s, ae_s, ae'\ \mathcal{S} \vdash ae \Rrightarrow_{ex} ae_s, t_s\ \wedge\ \mathcal{S} \vdash ae \equiv ae'\ \exists e.\ ae \Rrightarrow_r e\ \wedge\ \sigma_v, \sigma_{id} \vdash e \Rrightarrow_s \_, \_, ae'$$

$$\implies \exists ae'_s, t'_s, p_s, e_s.\ \mathcal{S} \vdash ae, ae'_s, t'_s \Rrightarrow_{st} ae'\ \wedge\ t_s \Rrightarrow_r p_s\ \wedge\ ae_s \Rrightarrow_r e_s$$

$$\wedge\ \sigma_v, \sigma_{id} \vdash p_s; \mathsf{assume}\ z = e_s \Rrightarrow_s t'_s; \mathsf{assume}\ z = ae'_s$$

Case 3: $ae = ((ae_1\ ae_2)aa : v)\#id$ and $ID(ae_1) \in \mathcal{S}$

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash ((ae_1\ ae_2)aa : v)\#id \Rightarrow_{ex} ((ae_s^1\ ae_s^3)aa : v)\#id, t_s$$

Then $ae_s = ((ae_s^1\ ae_s^3)aa : v)\#id$, $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$, $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^3, t_s^3$, and $t_s = t_s^1; t_s^3$.

By assumption

$$\mathcal{S} \vdash ae \equiv ae'$$

By definition of $\equiv$

$$\mathcal{S} \vdash ((ae_1\ ae_2)aa : v)\#id \equiv ((ae_1'\ ae_2')aa' : v')\#id'$$

Then $ae' = ((ae_1'\ ae_2')aa' : v')\#id'$, $\mathcal{S} \vdash ae_1 \equiv ae_1'$, $\mathcal{S} \vdash ae_2 \equiv ae_2'$.

By assumption

$$\exists e.\ ae \Rightarrow_r e$$

By definition of $\Rightarrow_r$

$$((ae_1\ ae_2)aa : v)\#id \Rightarrow_r (e_1\ e_2)$$

Then $e = (e_1\ e_2)$, $ae_1 \Rightarrow_r e_1$, and $ae_2 \Rightarrow_r e_2$.

By assumption

$$\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash (e_1\ e_2) \Rightarrow_s \_, \_, ((ae_1'\ ae_2')aa' : v')\#id'$$

Then $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1'$ and $\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae_2'$.

If $\mathcal{V}(ae_1') = \langle \lambda.x\ e', \sigma_v', \sigma_{id}' \rangle$, $\sigma_v'[y \to \mathcal{V}(ae_2')], \sigma_{id}'[y \to ID(ae_2')] \vdash e'[y/x] \Rightarrow_s \_, \_, ae_e$,

$aa = y = ae_e$, else $aa = \perp$.

By induction hypothesis

$$\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1 \;\wedge\; \mathcal{S} \vdash ae_1 \equiv ae_1' \;\wedge\; ae_1 \Rightarrow_r e_1 \;\wedge \sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1'$$

$$\implies \mathcal{S} \vdash ae_1, ae_s^2, t_s^2 \Rightarrow_{st} ae_1' \;\wedge\; t_s^1 \Rightarrow_r p_s^1 \;\wedge\; ae_s^1 \Rightarrow_r e_s^1$$

$$\wedge\; \sigma_v, \sigma_{id} \vdash p_s^1; \textsf{assume } z = e_s^1 \Rightarrow_s t_s^2; \textsf{assume } z = ae_s^2$$

Because $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$, $\mathcal{S} \vdash ae_1 \equiv ae_1'$, $ae_1 \Rightarrow_r e_1$, and $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1'$,

$$\mathcal{S} \vdash ae_1, ae_s^2, t_s^2 \Rightarrow_{st} ae_1' \;\wedge\; t_s^1 \Rightarrow_r p_s^1 \;\wedge\; ae_s^1 \Rightarrow_r e_s^1$$

$$\wedge\; \sigma_v, \sigma_{id} \vdash p_s^1; \textsf{assume } z = e_s^1 \Rightarrow_s t_s^2; \textsf{assume } z = ae_s^2$$

By induction hypothesis

$$\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^3, t_s^3 \;\wedge\; \mathcal{S} \vdash ae_2 \equiv ae_2' \;\wedge\; ae_2 \Rightarrow_r e_2 \;\wedge \sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae_2'$$

$$\implies \mathcal{S} \vdash ae_2, ae_s^4, t_s^4 \Rightarrow_{st} ae_2' \;\wedge\; t_s^3 \Rightarrow_r p_s^2 \;\wedge\; ae_s^3 \Rightarrow_r e_s^2$$

$$\wedge\; \sigma_v, \sigma_{id} \vdash p_s^2; \textsf{assume } z = e_s^2 \Rightarrow_s t_s^4; \textsf{assume } z = ae_s^4$$

Because $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^3, t_s^3$, $\mathcal{S} \vdash ae_2 \equiv ae_2'$, $ae_2 \Rightarrow_r e_2$, and $\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae_2'$,

$$\mathcal{S} \vdash ae_2, ae_s^4, t_s^4 \Rightarrow_{st} ae_2' \;\wedge\; t_s^3 \Rightarrow_r p_s^2 \;\wedge\; ae_s^3 \Rightarrow_r e_s^2$$

$$\wedge\; \sigma_v, \sigma_{id} \vdash p_s^2; \textsf{assume } z = e_s^2 \Rightarrow_s t_s^4; \textsf{assume } z = ae_s^4$$

Consider $ae_s' = ((ae_s^2\ ae_s^4)aa' : v')\#id'$, $t_s' = t_s^2; t_s^4$, $p_s = p_s^1; p_s^2$, and $e_s = (e_s^1\ e_s^2)$. Because $\mathcal{S} \vdash ae_1, ae_s^2, t_s^2 \Rightarrow_{st} ae_1'$, $\mathcal{S} \vdash ae_2, ae_s^4, t_s^4 \Rightarrow_{st} ae_2'$, and $id_e \in \mathcal{S}$, the definition of $\Rightarrow_{st}$ implies

$$\mathcal{S} \vdash ((ae_1\ ae_2)aa : v)\#id, ((ae_s^2\ ae_s^4)aa' : v')\#id', t_s' \Rightarrow_{st} ((ae_1'\ ae_2')aa' : v')\#id'$$

Therefore

$$\mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae'$$

Because $t^1_s \Rightarrow_r p^1_s$ and $t^3_s \Rightarrow_r p^2_s$, definition of $\Rightarrow_r$ implies

$$t^1_s; t^3_s \Rightarrow_r p^1_s; p^2_s$$

Therefore

$$t_s \Rightarrow_r p_s$$

Because $ae^3_s \Rightarrow_r e^2_s$ and $ae^1_s \Rightarrow_r e^1_s$, the definition of $\Rightarrow_r$ implies

$$((ae^1_s\ ae^3_s)aa : v)\#id \implies (e^1_s\ e^2_s)$$

Therefore

$$ae_s \Rightarrow_r e_s$$

Because $\sigma_v, \sigma_{id} \vdash p^1_s; \textsf{assume}\ z = e^1_s \Rightarrow_s t^2_s; \textsf{assume}\ z = ae^2_s$, $\sigma_v, \sigma_{id} \vdash p^2_s; \textsf{assume}\ z = e^2_s \Rightarrow_s t^4_s; \textsf{assume}\ z = ae^4_s$, and all variable names introduced by t^2_s do not conflict with variable names in ae^4_s and t^4_s (Observation 1), the definition of $\Rightarrow_s$ implies

$$\sigma_v, \sigma_{id} \vdash p_s; \textsf{assume}\ z = (e^1_s\ e^2_s) \Rightarrow_s t'_s; \textsf{assume}\ z = (ae^2_s\ ae^4_s)aa'$$

Therefore

$$\sigma_v, \sigma_{id} \vdash p_s; \textsf{assume}\ z = e_s \Rightarrow_s t'_s; \textsf{assume}\ z = ae'_s$$

Therefore when $ae = ((ae_1\ ae_2)aa : v)\#id$ and $id_e \in \mathcal{S}$, and assuming induction hypothesis,

$$\forall t_s, ae_s, ae'\ \mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s\ \wedge\ \mathcal{S} \vdash ae \equiv ae'\ \exists e.\ ae \Rightarrow_r e\ \wedge\ \sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

$$\implies \exists ae'_s, t'_s, p_s, e_s.\ \mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae'\ \wedge\ t_s \Rightarrow_r p_s\ \wedge\ ae_s \Rightarrow_r e_s$$

$$\wedge\ \sigma_v, \sigma_{id} \vdash p_s; \textsf{assume}\ z = e_s \Rightarrow_s t'_s; \textsf{assume}\ z = ae'_s$$

Case 4: $ae = ((ae_1\ ae_2)\ \bot\!:v)\#id$ and $ID(ae_1) \notin \mathcal{S}$

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash ((ae_1\ ae_2)\ \bot\!:v)\#id \Rightarrow_{ex} ((ae_s^1\ ae_s^3)\ \bot\!:v)\#id, t_s$$

Then $ae_s = ((ae_s^1\ ae_s^3)\ \bot\!:v)\#id$, $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$, $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^3, t_s^3$, and $t_s = t_s^1; t_s^3$.

By assumption

$$\mathcal{S} \vdash ae \equiv ae'$$

By definition of $\equiv$

$$\mathcal{S} \vdash ((ae_1\ ae_2)\ \bot\!:v)\#id \equiv ((ae_1'\ ae_2')\ \bot\!:v')\#id'$$

Then $ae' = ((ae_1'\ ae_2')\ \bot\!:v')\#id'$, $\mathcal{S} \vdash ae_1 \equiv ae_1'$, $\mathcal{S} \vdash ae_2 \equiv ae_2'$.

By assumption

$$\exists e.\ ae \Rightarrow_r e$$

By definition of $\Rightarrow_r$

$$((ae_1\ ae_2)\ \bot\!:v)\#id \Rightarrow_r (e_1\ e_2)$$

Then $e = (e_1\ e_2)$, $ae_1 \Rightarrow_r e_1$, and $ae_2 \Rightarrow_r e_2$.

By assumption

$$\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash (e_1\ e_2) \Rightarrow_s \_, \_, ((ae_1'\ ae_2')\ \bot\!:v')\#id'$$

Then $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1'$ and $\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae_2'$.

By induction hypothesis

$$\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1 \;\wedge\; \mathcal{S} \vdash ae_1 \equiv ae_1' \;\wedge\; ae_1 \Rightarrow_r e_1 \;\wedge \sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1'$$

$$\implies \mathcal{S} \vdash ae_1, ae_s^2, t_s^2 \Rightarrow_{st} ae_1' \;\wedge\; t_s^1 \Rightarrow_r p_s^1 \;\wedge\; ae_s^1 \Rightarrow_r e_s^1$$

$$\wedge\; \sigma_v, \sigma_{id} \vdash p_s^1; \textsf{assume}\ z = e_s^1 \Rightarrow_s t_s^2; \textsf{assume}\ z = ae_s^2$$

Because $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$, $\mathcal{S} \vdash ae_1 \equiv ae_1'$, $ae_1 \Rightarrow_r e_1$, and $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae_1'$,

$$\mathcal{S} \vdash ae_1, ae_s^2, t_s^2 \Rightarrow_{st} ae_1' \;\wedge\; t_s^1 \Rightarrow_r p_s^1 \;\wedge\; ae_s^1 \Rightarrow_r e_s^1$$

$$\wedge\; \sigma_v, \sigma_{id} \vdash p_s^1; \textsf{assume}\ z = e_s^1 \Rightarrow_s t_s^2; \textsf{assume}\ z = ae_s^2$$

By induction hypothesis

$$\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^3, t_s^3 \;\wedge\; \mathcal{S} \vdash ae_2 \equiv ae_2' \;\wedge\; ae_2 \Rightarrow_r e_2 \;\wedge \sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae_2'$$

$$\implies \mathcal{S} \vdash ae_2, ae_s^4, t_s^4 \Rightarrow_{st} ae_2' \;\wedge\; t_s^3 \Rightarrow_r p_s^2 \;\wedge\; ae_s^3 \Rightarrow_r e_s^2$$

$$\wedge\; \sigma_v, \sigma_{id} \vdash p_s^2; \textsf{assume}\ z = e_s^2 \Rightarrow_s t_s^4; \textsf{assume}\ z = ae_s^4$$

Because $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^3, t_s^3$, $\mathcal{S} \vdash ae_2 \equiv ae_2'$, $ae_2 \Rightarrow_r e_2$, and $\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae_2'$,

$$\mathcal{S} \vdash ae_2, ae_s^4, t_s^4 \Rightarrow_{st} ae_2' \;\wedge\; t_s^3 \Rightarrow_r p_s^2 \;\wedge\; ae_s^3 \Rightarrow_r e_s^2$$

$$\wedge\; \sigma_v, \sigma_{id} \vdash p_s^2; \textsf{assume}\ z = e_s^2 \Rightarrow_s t_s^4; \textsf{assume}\ z = ae_s^4$$

Consider $ae_s' = ((ae_s^2\ ae_s^4) \perp\!: v')\#id'$, $t_s' = t_s^2; t_s^4$, $p_s = p_s^1; p_s^2$, and $e_s = (e_s^1\ e_s^2)$. Because $\mathcal{S} \vdash ae_1, ae_s^2, t_s^2 \Rightarrow_{st} ae_1'$, $\mathcal{S} \vdash ae_2, ae_s^4, t_s^4 \Rightarrow_{st} ae_2'$, and $id_e \in \mathcal{S}$, the definition of $\Rightarrow_{st}$ implies

$$\mathcal{S} \vdash ((ae_1\ ae_2) \perp\!: v)\#id, ((ae_s^2\ ae_s^4) \perp\!: v')\#id', t_s' \Rightarrow_{st} ((ae_1'\ ae_2') \perp\!: v')\#id'$$

Therefore

$$\mathcal{S} \vdash ae, ae'_s, t'_s \Rrightarrow_{st} ae'$$

Because $t^1_s \Rrightarrow_r p^1_s$ and $t^3_s \Rrightarrow_r p^2_s$, definition of $\Rrightarrow_r$ implies

$$t^1_s; t^3_s \Rrightarrow_r p^1_s; p^2_s$$

Therefore

$$t_s \Rrightarrow_r p_s$$

Because $ae^3_s \Rrightarrow_r e^2_s$ and $ae^1_s \Rrightarrow_r e^1_s$, the definition of $\Rrightarrow_r$ implies

$$((ae^1_s\ ae^3_s)\ \bot\colon v)\#id \implies (e^1_s\ e^2_s)$$

Therefore

$$ae_s \Rrightarrow_r e_s$$

Because $\sigma_v, \sigma_{id} \vdash p^1_s; \textsf{assume}\ z = e^1_s \Rrightarrow_s t^2_s; \textsf{assume}\ z = ae^2_s$, $\sigma_v, \sigma_{id} \vdash p^2_s; \textsf{assume}\ z = e^2_s \Rrightarrow_s t^4_s; \textsf{assume}\ z = ae^4_s$, and all variable names introduced by t^2_s do not conflict with variable names in ae^4_s and t^4_s (Observation 1), the definition of $\Rrightarrow_s$ implies

$$\sigma_v, \sigma_{id} \vdash p_s; \textsf{assume}\ z = (e^1_s\ e^2_s) \Rrightarrow_s t'_s; \textsf{assume}\ z = (ae^2_s\ ae^4_s)\ \bot$$

Therefore

$$\sigma_v, \sigma_{id} \vdash p_s; \textsf{assume}\ z = e_s \Rrightarrow_s t'_s; \textsf{assume}\ z = ae'_s$$

Therefore when $ae = ((ae_1\ ae_2)\ \bot\colon v)\#id$ and $id_e \notin \mathcal{S}$, and assuming induction hypothesis,

$$\forall t_s, ae_s, ae'\ \mathcal{S} \vdash ae \Rrightarrow_{ex} ae_s, t_s\ \wedge\ \mathcal{S} \vdash ae \equiv ae'\ \exists e.\ ae \Rrightarrow_r e\ \wedge\ \sigma_v, \sigma_{id} \vdash e \Rrightarrow_s \_, \_, ae'$$

$$\implies \exists ae'_s, t'_s, p_s, e_s.\ \mathcal{S} \vdash ae, ae'_s, t'_s \Rrightarrow_{st} ae'\ \wedge\ t_s \Rrightarrow_r p_s\ \wedge\ ae_s \Rrightarrow_r e_s$$

$$\wedge\ \sigma_v, \sigma_{id} \vdash p_s; \textsf{assume}\ z = e_s \Rrightarrow_s t'_s; \textsf{assume}\ z = ae'_s$$

Case 5: $ae = ((ae_1\ ae_2)y = ae_3 : v)\#id$ and $ID(ae_1) \notin \mathcal{S}$

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash ((ae_1\ ae_2)y = ae_3 : v)\#id \Rightarrow_{ex} (ae_s^5 : v)\#id, t_s$$

Then $ae_s = (ae_s^5 : v)\#id$, $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$, $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^3, t_s^3$, $\mathcal{S} \vdash ae_3 \rightarrow_{ex} ae_s^5, t_s^5$, and $t_s = t_s^1$; assume $x = ae_s^1; t_s^3$; assume $y = ae_s^3; t_s^5$.

By assumption

$$\mathcal{S} \vdash ae \equiv ae'$$

By definition of $\equiv$

$$\mathcal{S} \vdash ((ae_1\ ae_2)y = ae_3 : v)\#id \equiv ((ae_1'\ ae_2')y = ae_3' : v')\#id'$$

Then $ae' = ((ae_1'\ ae_2')y = ae_3' : v')\#id'$, $\mathcal{S} \vdash ae_1 \equiv ae_1'$, $\mathcal{S} \vdash ae_2 \equiv ae_2'$, and $\mathcal{S} \vdash ae_3 \equiv ae_3'$.

By assumption

$$\exists e.\ ae \Rightarrow_r e$$

By definition of $\Rightarrow_r$

$$((ae_1\ ae_2)y = ae_3 : v)\#id \Rightarrow_r (e_1\ e_2)$$

Then $e = (e_1\ e_2)$, $ae_1 \Rightarrow_r e_1$, and $ae_2 \Rightarrow_r e_2$.

By assumption

$$\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash (e_1\ e_2) \Rightarrow_s \_, \_, ((ae_1'\ ae_2')y = ae_3' : v')\#id'$$

Then $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae'_1$ and $\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae'_2$, $\mathcal{V}(ae'_1) = \langle \lambda.x\, e'_3, \sigma'_v, \sigma'_{id} \rangle$, $e_3 = e'_3[y/x]$, and $\sigma'_v[y \to \mathcal{V}(ae'_2)], \sigma'_{id}[y \to ID(ae'_2)] \vdash e_3 \Rightarrow_s \_, \_, ae'_3$.

By induction hypothesis

$$\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae^1_s, t^1_s \ \wedge\ \mathcal{S} \vdash ae_1 \equiv ae'_1 \ \wedge\ ae_1 \Rightarrow_r e_1 \ \wedge \sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae'_1$$

$$\Longrightarrow\ \mathcal{S} \vdash ae_1, ae^2_s, t^2_s \Rightarrow_{st} ae'_1 \ \wedge\ t^1_s \Rightarrow_r p^1_s \ \wedge\ ae^1_s \Rightarrow_r e^1_s$$

$$\wedge\ \sigma_v, \sigma_{id} \vdash p^1_s; \textsf{assume}\ z = e^1_s \Rightarrow_s t^2_s; \textsf{assume}\ z = ae^2_s$$

Because $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae^1_s, t^1_s$, $\mathcal{S} \vdash ae_1 \equiv ae'_1$, $ae_1 \Rightarrow_r e_1$, and $\sigma_v, \sigma_{id} \vdash e_1 \Rightarrow_s \_, \_, ae'_1$,

$$\mathcal{S} \vdash ae_1, ae^2_s, t^2_s \Rightarrow_{st} ae'_1 \ \wedge\ t^1_s \Rightarrow_r p^1_s \ \wedge\ ae^1_s \Rightarrow_r e^1_s$$

$$\wedge\ \sigma_v, \sigma_{id} \vdash p^1_s; \textsf{assume}\ z = e^1_s \Rightarrow_s t^2_s; \textsf{assume}\ z = ae^2_s$$

By induction hypothesis

$$\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae^3_s, t^3_s \ \wedge\ \mathcal{S} \vdash ae_2 \equiv ae'_2 \ \wedge\ ae_2 \Rightarrow_r e_2 \ \wedge \sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae'_2$$

$$\Longrightarrow\ \mathcal{S} \vdash ae_2, ae^4_s, t^4_s \Rightarrow_{st} ae'_2 \ \wedge\ t^3_s \Rightarrow_r p^2_s \ \wedge\ ae^3_s \Rightarrow_r e^2_s$$

$$\wedge\ \sigma_v, \sigma_{id} \vdash p^2_s; \textsf{assume}\ z = e^2_s \Rightarrow_s t^4_s; \textsf{assume}\ z = ae^4_s$$

Because $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae^3_s, t^3_s$, $\mathcal{S} \vdash ae_2 \equiv ae'_2$, $ae_2 \Rightarrow_r e_2$, and $\sigma_v, \sigma_{id} \vdash e_2 \Rightarrow_s \_, \_, ae'_2$,

$$\mathcal{S} \vdash ae_2, ae^4_s, t^4_s \Rightarrow_{st} ae'_2 \ \wedge\ t^3_s \Rightarrow_r p^2_s \ \wedge\ ae^3_s \Rightarrow_r e^2_s$$

$$\wedge\ \sigma_v, \sigma_{id} \vdash p^2_s; \textsf{assume}\ z = e^2_s \Rightarrow_s t^4_s; \textsf{assume}\ z = ae^4_s$$

By induction hypothesis

$$\mathcal{S} \vdash ae_3 \Rightarrow_{ex} ae^5_s, t^5_s \ \wedge\ \mathcal{S} \vdash ae_3 \equiv ae'_3 \ \wedge\ ae_3 \Rightarrow_r e_3 \ \wedge \sigma_v, \sigma_{id} \vdash e_3 \Rightarrow_s \_, \_, ae'_3$$

$$\Longrightarrow\ \mathcal{S} \vdash ae_3, ae^6_s, t^6_s \Rightarrow_{st} ae'_3 \ \wedge\ t^5_s \Rightarrow_r p^3_s \ \wedge\ ae^5_s \Rightarrow_r e^3_s$$

$$\wedge\ \sigma_v, \sigma_{id} \vdash p_s^3; \textsf{assume } z = e_s^3 \Rightarrow_s t_s^6; \textsf{assume } z = ae_s^6$$

Because $\mathcal{S} \vdash ae_3 \Rightarrow_{ex} ae_s^5, t_s^5$, $\mathcal{S} \vdash ae_3 \equiv ae_3'$, $ae_3 \Rightarrow_r e_3$, and $\sigma_v, \sigma_{id} \vdash e_3 \Rightarrow_s \_, \_, ae_3'$,

$$\mathcal{S} \vdash ae_3, ae_s^6, t_s^6 \Rightarrow_{st} ae_3' \ \wedge\ t_s^5 \Rightarrow_r p_s^3 \ \wedge\ ae_s^5 \Rightarrow_r e_s^3$$

$$\wedge\ \sigma_v, \sigma_{id} \vdash p_s^3; \textsf{assume } z = e_s^3 \Rightarrow_s t_s^6; \textsf{assume } z = ae_s^6$$

Consider $ae_s' = (ae_s^6 : v')\#id'$, $t_s' = t_s^2; \textsf{assume } x = ae_s^2; t_s^4; \textsf{assume } y = ae_s^4; t_s^6$, $p_s = p_s^1; \textsf{assume } x = e_s^1; p_s^2; \textsf{assume } y = e_s^2; p_s^3$, and $e_s = e_s^3$. Because $\mathcal{S} \vdash ae_1, ae_s^2, t_s^2 \Rightarrow_{st} ae_1'$, $\mathcal{S} \vdash ae_2, ae_s^4, t_s^4 \Rightarrow_{st} ae_2'$, $\mathcal{S} \vdash ae_3, ae_s^6, t_s^6 \Rightarrow_{st} ae_3'$, the definition of $\Rightarrow_{st}$ implies

$$\mathcal{S} \vdash ((ae_1\ ae_2)y = ae_3 : v)\#id, (ae_s^6 : v')\#id', t_s' \Rightarrow_{st} ((ae_1'\ ae_2')y = ae_3' : v')\#id'$$

Therefore

$$\mathcal{S} \vdash ae, ae_s', t_s' \Rightarrow_{st} ae'$$

Because $t_s^1 \Rightarrow_r p_s^1$, $t_s^3 \Rightarrow_r p_s^2$, $t_s^5 \Rightarrow_r p_s^3$, $ae_s^1 \Rightarrow_r e_s^1$, and $ae_s^3 \Rightarrow_r e_s^2$, definition of $\Rightarrow_r$ implies

$$t_s^1; \textsf{assume } x = ae_s^1; t_s^3; \textsf{assume } y = ae_s^3 t_s^5 \Rightarrow_r p_s^1; \textsf{assume } x = e_s^1; p_s^2; \textsf{assume } y = e_s^2; p_s^3$$

Therefore

$$t_s \Rightarrow_r p_s$$

Because $ae_s^5 \Rightarrow_r e_s^3$, the definition of $\Rightarrow_r$ implies

$$(ae_s^5 : v)\#id \implies e_s^3$$

Therefore

$$ae_s \Rightarrow_r e_s$$

Because $\sigma_v, \sigma_{id} \vdash p_s^1; \textsf{assume } z = e_s^1 \Rightarrow_s t_s^2; \textsf{assume } z = ae_s^2$, $\sigma_v, \sigma_{id} \vdash p_s^2; \textsf{assume } z = e_s^2 \Rightarrow_s t_s^4; \textsf{assume } z = ae_s^4$, and all variable names introduced by t_s^2 do not conflict with variable names in ae_s^4 and t_s^4 (Observation 1), the definition of $\Rightarrow_s$ implies

$$\sigma_v, \sigma_{id} \vdash p_s; \textsf{assume}\ z = e_s^3 \Rightarrow_s t'_s; \textsf{assume}\ z = ae_s^6$$

Therefore

$$\sigma_v, \sigma_{id} \vdash p_s; \textsf{assume}\ z = e_s \Rightarrow_s t'_s; \textsf{assume}\ z = ae'_s$$

Therefore when $ae = ((ae_1\ ae_2)y = ae_3 : v)\#id$ and $id_e \notin \mathcal{S}$, and assuming induction hypothesis,

$$\forall t_s, ae_s, ae'\ \mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s\ \wedge\ \mathcal{S} \vdash ae \equiv ae'\ \exists e.\ ae \Rightarrow_r e\ \wedge\ \sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

$$\implies \exists ae'_s, t'_s, p_s, e_s.\ \mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae'\ \wedge\ t_s \Rightarrow_r p_s\ \wedge\ ae_s \Rightarrow_r e_s$$

$$\wedge\ \sigma_v, \sigma_{id} \vdash p_s; \textsf{assume}\ z = e_s \Rightarrow_s t'_s; \textsf{assume}\ z = ae'_s$$

Because we have covered all cases, using induction, the following statement is true

$$\forall t_s, ae_s, ae'\ \mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s\ \wedge\ \mathcal{S} \vdash ae \equiv ae'\ \exists e.\ ae \Rightarrow_r e\ \wedge\ \sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

$$\implies \exists ae'_s, t'_s, p_s, e_s.\ \mathcal{S} \vdash ae, ae'_s, t'_s \Rightarrow_{st} ae'\ \wedge\ t_s \Rightarrow_r p_s\ \wedge\ ae_s \Rightarrow_r e_s$$

$$\wedge\ \sigma_v, \sigma_{id} \vdash p_s; \textsf{assume}\ z = e_s \Rightarrow_s t'_s; \textsf{assume}\ z = ae'_s$$

$\square$

Lemma 7. *For any two traces t, t', a valid subproblem $\mathcal{S}$ on t and a subtrace $\mathcal{S} \vdash t \Rightarrow_{ex} t_s$*

$$\mathcal{S} \vdash t \equiv t' \wedge t \Rightarrow_r p \wedge \sigma_v, \sigma_{id} \vdash p \Rightarrow_s t'$$

implies there exists a subtrace t'_s such that

$$\exists\ p_s. t_s \Rightarrow_r p_s \wedge \sigma_v, \sigma_{id} \vdash p_s \Rightarrow_s t'_s \wedge \mathcal{S} \vdash t, t'_s \Rightarrow_{st} t'$$

Proof. Proof by Induction

Base Case: $t = \emptyset$

By assumption

$$\mathcal{S} \vdash t \Rightarrow_{ex} t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash \emptyset \Rightarrow_{ex} \emptyset$$

Then $t_s = \emptyset$.

By assumption

$$\mathcal{S} \vdash t \equiv t'$$

By definition of $\equiv$

$$\mathcal{S} \vdash \emptyset \equiv \emptyset$$

Then $t' = \emptyset$.

By assumption

$$t \Rightarrow_r p$$

By definition of $\Rightarrow_r$

$$\emptyset \Rightarrow_r \emptyset$$

Then $p = \emptyset$.

By assumption

$$\sigma_v, \sigma_{id} \vdash p \Rightarrow_s t'$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash \emptyset \Rightarrow_s \emptyset$$

Consider $t'_s = \emptyset$, $p_s = \emptyset$. By definition of $\Rightarrow_r$

$$\emptyset \Rightarrow_r \emptyset$$

Therefore

$$t_s \Rightarrow_r p_s$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash \emptyset \Rightarrow_s \emptyset$$

Therefore

$$\sigma_v, \sigma_{id} \vdash p_s \Rightarrow_s t'_s$$

By definition of $\Rightarrow_{st}$

$$\mathcal{S} \vdash \emptyset, \emptyset \Rightarrow_{st} \emptyset$$

Therefore

$$\mathcal{S} \vdash t, t'_s \Rightarrow_{st} t'$$

Therefore when $t = \emptyset$,

$$\forall\ t', t_s, p.\ \mathcal{S} \vdash t \Rightarrow_{ex} t_s\ \wedge\ \mathcal{S} \vdash t \equiv t'\ \wedge t \Rightarrow_r p \wedge \sigma_v, \sigma_{id} \vdash p \Rightarrow_s t'$$

$$\implies\ \exists\ t'_s, p_s . t_s \Rightarrow_r p_s\ \wedge\ \sigma_v, \sigma_{id} \vdash p_s \Rightarrow_s t'_s\ \wedge\ \mathcal{S} \vdash t, t'_s \Rightarrow_{st} t'$$

Induction Case:

Case 1: $t = \textsf{assume}\ x = ae; t_1$

By assumption

$$\mathcal{S} \vdash t \Rightarrow_{ex} t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash \textsf{assume}\ x = ae; t_1 \Rightarrow_{ex} t_s^1; \textsf{assume}\ x = ae_s; t_s^3$$

Then $t_s = t_s^1; \textsf{assume}\ x = ae_s; t_s^3$, $\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s^1$ and $\mathcal{S} \vdash t_1 \Rightarrow_{ex} t_s^3$.

By assumption

$$\mathcal{S} \vdash t \equiv t'$$

By definition of $\equiv$

$$\mathcal{S} \vdash \textsf{assume}\ x = ae; t_1 \equiv \textsf{assume}\ x = ae'; t'_1$$

Then $t' = \textsf{assume } x = ae'; t_1'$, $\mathcal{S} \vdash ae \equiv ae'$, and $\mathcal{S} \vdash t_1 \equiv t_1'$.

By assumption

$$t \Rrightarrow_r p$$

By definition of $\Rrightarrow_r$

$$\textsf{assume } x = ae; t_1 \Rrightarrow_r \textsf{assume } x = e; p_1$$

Then $p = \textsf{assume } x = e; p_1$, $ae \Rrightarrow_r e$, and $t_1 \Rrightarrow_r p_1$.

By assumption

$$\sigma_v, \sigma_{id} \vdash p \Rrightarrow_s t'$$

By definition of $\Rrightarrow_s$

$$\sigma_v, \sigma_{id} \vdash \textsf{assume } x = e; p_1 \Rrightarrow_s \textsf{assume } x = ae'; t_1'$$

Then $\sigma_v, \sigma_{id} \vdash e \Rrightarrow_s \_, \_, ae'$, $\sigma_v', \sigma_{id}' \vdash p_1 \Rrightarrow_s t_1'$, and $\sigma_v' = \sigma_v[x \to \mathcal{V}(ae')]$, $\sigma_{id}' = \sigma_{id}[x \to ID(ae')]$.

By induction hypothesis

$$\mathcal{S} \vdash t_1 \Rrightarrow_{ex} t_s^3 \;\wedge\; \mathcal{S} \vdash t_1 \equiv t_1' \;\wedge t_1 \Rrightarrow_r p_1 \wedge \sigma_v', \sigma_{id}' \vdash p_1 \Rrightarrow_s t_1'$$

$$\implies \exists\, t_s^4, p_s^2 . t_s^3 \Rrightarrow_r p_s^2 \;\wedge\; \sigma_v', \sigma_{id}' \vdash p_s^2 \Rrightarrow_s t_s^4 \;\wedge\; \mathcal{S} \vdash t_1, t_s^4 \Rrightarrow_{st} t_1'$$

Because $\mathcal{S} \vdash t_1 \Rrightarrow_{ex} t_s^3$, $\mathcal{S} \vdash t_1 \equiv t_1'$, $t_1 \Rrightarrow_r p_1$, and $\sigma_v', \sigma_{id}' \vdash p_1 \Rrightarrow_s t_1'$,

$$t_s^3 \Rrightarrow_r p_s^2 \;\wedge\; \sigma_v', \sigma_{id}' \vdash p_s^2 \Rrightarrow_s t_s^4 \;\wedge\; \mathcal{S} \vdash t_1, t_s^4 \Rrightarrow_{st} t_1'$$

By induction hypothesis

$$\forall t_s^1, ae_s, ae'\; \mathcal{S} \vdash ae \Rrightarrow_{ex} ae_s, t_s^1 \;\wedge\; \mathcal{S} \vdash ae \equiv ae'\; \exists e.\; ae \Rrightarrow_r e \;\wedge\; \sigma_v, \sigma_{id} \vdash e \Rrightarrow_s \_, \_, ae'$$

$$\implies \exists ae_s', t_s^2, p_s^1, e_s.\; \mathcal{S} \vdash ae, ae_s', t_s^2 \Rrightarrow_{st} ae' \;\wedge\; t_s^1 \Rrightarrow_r p_s^1 \;\wedge\; ae_s \Rrightarrow_r e_s$$

$$\wedge\ \sigma_v, \sigma_{id} \vdash p_s^1; \textsf{assume } z = e_s \Rightarrow_s t_s^2; \textsf{assume } z = ae'_s$$

Because $\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s^1$, $\mathcal{S} \vdash ae \equiv ae'$, $ae \Rightarrow_r e$, and $\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$,

$$\mathcal{S} \vdash ae, ae'_s, t_s^2 \Rightarrow_{st} ae' \ \wedge\ t_s^1 \Rightarrow_r p_s^1 \ \wedge\ ae_s \Rightarrow_r e_s$$

$$\wedge\ \sigma_v, \sigma_{id} \vdash p_s^1; \textsf{assume } z = e_s \Rightarrow_s t_s^2; \textsf{assume } z = ae'_s$$

Because $\mathcal{S} \vdash ae, ae'_s, t_s^2 \Rightarrow_{st} ae'$ and $\mathcal{S} \vdash t_1, t_s^4 \Rightarrow_{st} t'_1$, the definition of $\Rightarrow_{st}$ implies

$$\mathcal{S} \vdash \textsf{assume } x = ae; t_1, t_s^2; \textsf{assume } x = ae'_s; t_s^4 \Rightarrow_{st} \textsf{assume } x = ae'; t'_1$$

Therefore

$$\mathcal{S} \vdash t, t'_s \Rightarrow_{st} t'$$

Because $t_s^3 \Rightarrow_r p_s^2$, $t_s^1 \Rightarrow_r p_s^1$, and $ae_s \Rightarrow_r e_s$, the definition of $\Rightarrow_r$ implies

$$t_s^1; \textsf{assume } x = ae_s; t_s^3 \Rightarrow_r p_s^1; \textsf{assume } x = e_s; p_s^2$$

Therefore

$$t \Rightarrow_r p$$

Because $\sigma_v, \sigma_{id} \vdash p_s^1; \textsf{assume } z = e_s \Rightarrow_s t_s^2; \textsf{assume } z = ae'_s$, $\sigma'_v, \sigma'_{id} \vdash p_s^2 \Rightarrow_s t_s^4$, all variable names introduced by t_s^2 do not conflict with variable names in t_s^4 (Observation 1), $\mathcal{V}(ae') = \mathcal{V}(ae'_s)$, and $ID(ae') = ID(ae'_s)$ (Observation 2), the definition of $\Rightarrow_s$ implies

$$\sigma_v, \sigma_{id} \vdash p_s^1; \textsf{assume } x = e_s; p_s^2 \Rightarrow_s t_s^2; \textsf{assume } x = ae'_s; t_s^4$$

Therefore

$$\sigma_v, \sigma_{id} \vdash p_s \Rightarrow_s t'_s$$

Therefore when $t = \textsf{assume}\ x = ae; t_1$, and assuming the induction hypothesis,

$$\forall\, t', t_s, p.\ \mathcal{S} \vdash t \Rightarrow_{ex} t_s \ \wedge\ \mathcal{S} \vdash t \equiv t' \ \wedge t \Rightarrow_r p \wedge \sigma_v, \sigma_{id} \vdash p \Rightarrow_s t'$$

$$\Longrightarrow\ \exists\, t'_s, p_s. t_s \Rightarrow_r p_s \ \wedge\ \sigma_v, \sigma_{id} \vdash p_s \Rightarrow_s t'_s \ \wedge\ \mathcal{S} \vdash t, t'_s \Rightarrow_{st} t'$$

Case 2: $t = \textsf{observe}(\textsf{Dist}(ae) = e_v); t_1$

By assumption

$$\mathcal{S} \vdash t \Rightarrow_{ex} t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash \textsf{observe}(\textsf{Dist}(ae) = e_v); t_1 \Rightarrow_{ex} t_s^1; \textsf{observe}(\textsf{Dist}(ae_s) = e_v); t_s^3$$

Then $t_s = t_s^1; \textsf{observe}(\textsf{Dist}(ae_s) = e_v); t_s^3$, $\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s^1$ and $\mathcal{S} \vdash t_1 \Rightarrow_{ex} t_s^3$.

By assumption

$$\mathcal{S} \vdash t \equiv t'$$

By definition of $\equiv$

$$\mathcal{S} \vdash \textsf{observe}(\textsf{Dist}(ae) = e_v); t_1 \equiv \textsf{observe}(\textsf{Dist}(ae') = e_v); t'_1$$

Then $t' = \textsf{observe}(\textsf{Dist}(ae') = e_v); t'_1$, $\mathcal{S} \vdash ae \equiv ae'$, and $\mathcal{S} \vdash t_1 \equiv t'_1$.

By assumption

$$t \Rightarrow_r p$$

By definition of $\Rightarrow_r$

$$\textsf{observe}(\textsf{Dist}(ae) = e_v); t_1 \Rightarrow_r \textsf{observe}(\textsf{Dist}(e) = e_v); p_1$$

Then $p = \textsf{observe}(\textsf{Dist}(e) = e_v); p_1$, $ae \Rightarrow_r e$, and $t_1 \Rightarrow_r p_1$.

By assumption

$$\sigma_v, \sigma_{id} \vdash p \Rightarrow_s t'$$

By definition of $\Rightarrow_s$

$$\sigma_v, \sigma_{id} \vdash \mathsf{observe}(\mathsf{Dist}(e) = e_v); p_1 \Rightarrow_s \mathsf{observe}(\mathsf{Dist}(ae') = e_v); t'_1$$

Then $\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$ and $\sigma_v, \sigma_{id} \vdash p_1 \Rightarrow_s t'_1$.

By induction hypothesis

$$\mathcal{S} \vdash t_1 \Rightarrow_{ex} t^3_s \;\wedge\; \mathcal{S} \vdash t_1 \equiv t'_1 \;\wedge t_1 \Rightarrow_r p_1 \wedge \sigma_v, \sigma_{id} \vdash p_1 \Rightarrow_s t'_1$$

$$\implies \exists\, t^4_s, p^2_s.t^3_s \Rightarrow_r p^2_s \;\wedge\; \sigma_v, \sigma_{id} \vdash p^2_s \Rightarrow_s t^4_s \;\wedge\; \mathcal{S} \vdash t_1, t^4_s \Rightarrow_{st} t'_1$$

Because $\mathcal{S} \vdash t_1 \Rightarrow_{ex} t^3_s$, $\mathcal{S} \vdash t_1 \equiv t'_1$, $t_1 \Rightarrow_r p_1$, and $\sigma_v, \sigma_{id} \vdash p_1 \Rightarrow_s t'_1$,

$$t^3_s \Rightarrow_r p^2_s \;\wedge\; \sigma_v, \sigma_{id} \vdash p^2_s \Rightarrow_s t^4_s \;\wedge\; \mathcal{S} \vdash t_1, t^4_s \Rightarrow_{st} t'_1$$

By induction hypothesis

$$\forall t^1_s, ae_s, ae'\; \mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t^1_s \;\wedge\; \mathcal{S} \vdash ae \equiv ae'\; \exists e.\; ae \Rightarrow_r e \;\wedge\; \sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$$

$$\implies \exists ae'_s, t^2_s, p^1_s, e_s.\; \mathcal{S} \vdash ae, ae'_s, t^2_s \Rightarrow_{st} ae' \;\wedge\; t^1_s \Rightarrow_r p^1_s \;\wedge\; ae_s \Rightarrow_r e_s$$

$$\wedge\; \sigma_v, \sigma_{id} \vdash p^1_s; \mathsf{assume}\ z = e_s \Rightarrow_s t^2_s; \mathsf{assume}\ z = ae'_s$$

Because $\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t^1_s$, $\mathcal{S} \vdash ae \equiv ae'$, $ae \Rightarrow_r e$, and $\sigma_v, \sigma_{id} \vdash e \Rightarrow_s \_, \_, ae'$,

$$\mathcal{S} \vdash ae, ae'_s, t^2_s \Rightarrow_{st} ae' \;\wedge\; t^1_s \Rightarrow_r p^1_s \;\wedge\; ae_s \Rightarrow_r e_s$$

$$\wedge\; \sigma_v, \sigma_{id} \vdash p^1_s; \mathsf{assume}\ z = e_s \Rightarrow_s t^2_s; \mathsf{assume}\ z = ae'_s$$

Because $\mathcal{S} \vdash ae, ae'_s, t^2_s \Rightarrow_{st} ae'$ and $\mathcal{S} \vdash t_1, t^4_s \Rightarrow_{st} t'_1$, the definition of $\Rightarrow_{st}$ implies

$$\mathcal{S} \vdash \mathsf{observe}(\mathsf{Dist}(ae) = e_v); t_1, t^2_s; \mathsf{observe}(\mathsf{Dist}(ae_s) = e_v); t^4_s \Rightarrow_{st} \mathsf{observe}(\mathsf{Dist}(ae') = e_v); t'_1$$

Therefore

$$\mathcal{S} \vdash t, t'_s \Rightarrow_{st} t'$$

Because $t^3_s \Rightarrow_r p^2_s$, $t^1_s \Rightarrow_r p^1_s$, and $ae_s \Rightarrow_r e_s$, the definition of $\Rightarrow_r$ implies

$$t^1_s; \mathsf{observe}(\mathsf{Dist}(ae_s) = e_v); t^3_s \Rightarrow_r p^1_s; \mathsf{observe}(\mathsf{Dist}(e_s) = e_v); p^2_s$$

Therefore

$$t \Rightarrow_r p$$

Because $\sigma_v, \sigma_{id} \vdash p^1_s; \mathsf{assume}\ z = e_s \Rightarrow_s t^2_s; \mathsf{assume}\ z = ae'_s$, $\sigma_v, \sigma_{id} \vdash p^2_s \Rightarrow_s t^4_s$, and all variable names introduced by t^2_s do not conflict with variable names in t^4_s (Observation 1), the definition of $\Rightarrow_s$ implies

$$\sigma_v, \sigma_{id} \vdash p^1_s; \mathsf{observe}(\mathsf{Dist}(e_s) = e_v); p^2_s \Rightarrow_s t^2_s; \mathsf{observe}(\mathsf{Dist}(ae'_s) = e_v); t^4_s$$

Therefore

$$\sigma_v, \sigma_{id} \vdash p_s \Rightarrow_s t'_s$$

Therefore when $t = \mathsf{observe}(\mathsf{Dist}(ae) = e_v); t_1$, and assuming the induction hypothesis,

$$\forall\ t', t_s, p.\ \mathcal{S} \vdash t \Rightarrow_{ex} t_s\ \wedge\ \mathcal{S} \vdash t \equiv t'\ \wedge t \Rightarrow_r p \wedge \sigma_v, \sigma_{id} \vdash p \Rightarrow_s t'$$

$$\Longrightarrow\ \exists\ t'_s, p_s. t_s \Rightarrow_r p_s\ \wedge\ \sigma_v, \sigma_{id} \vdash p_s \Rightarrow_s t'_s\ \wedge\ \mathcal{S} \vdash t, t'_s \Rightarrow_{st} t'$$

All cases have been covered therefore by induction the following statement is true.

$$\forall\ t', t_s, p.\ \mathcal{S} \vdash t \Rightarrow_{ex} t_s\ \wedge\ \mathcal{S} \vdash t \equiv t'\ \wedge t \Rightarrow_r p \wedge \sigma_v, \sigma_{id} \vdash p \Rightarrow_s t'$$

$$\Longrightarrow\ \exists\ t'_s, p_s. t_s \Rightarrow_r p_s\ \wedge\ \sigma_v, \sigma_{id} \vdash p_s \Rightarrow_s t'_s\ \wedge\ \mathcal{S} \vdash t, t'_s \Rightarrow_{st} t'$$

□

Theorem 2. *Given a valid trace t and a valid subproblem $\mathcal{S}$ of t and a subtrace $t_s = \mathsf{ExtractTrace}(t, \mathcal{S})$. For all possible traces t', $t' \in \mathsf{Traces}(\mathsf{Program}(t)) \wedge \mathcal{S} \vdash t \equiv t'$*

implies there exists a subtrace t'_s such that

- $t'_s \in \mathsf{Traces}(\mathsf{Program}(t_s))$
- $t' = \mathsf{StitchTrace}(t, t'_s, \mathcal{S})$

Proof. From definitions of Traces, Program, ExtractTrace, and StitchTrace, given a trace t and a valid subproblem $\mathcal{S}$

$$\exists\, p.\mathcal{S} \vdash t \Rightarrow_{ex} t_s \;\wedge\; \mathcal{S} \vdash t \equiv t' \;\wedge\; \emptyset, \emptyset \vdash p \Rightarrow_s t \;\wedge\; \emptyset, \emptyset \vdash p \Rightarrow_s t'$$

$$\implies \exists\, t'_s.\mathcal{S} \vdash t, t'_s \Rightarrow_{st} t' \;\wedge\; t_s \Rightarrow_r p_s \;\wedge\; \emptyset, \emptyset \vdash p_s \Rightarrow_s t_s$$

Because Lemma 7 is true for all environments $\sigma_v, \sigma_{id}, \sigma'_v$, and σ'_{id}. The theorem is equivalent to the lemma, but with $\sigma_v, \sigma_{id}, \sigma'_v$, and σ'_{id} set to $\emptyset$. □

Chapter 4

Convergence of Stochastic Alternating Class Kernels

In this chapter, I introduce the concept of class functions and class kernels which I use to prove the convergence of hybrid inference algorithms based on asymptotically converging MCMC-algorithms. I present key definitions, theorems, and lemmas required to prove these results.

4.1 Preliminaries

I begin by introducing some key measure theory definitions required for my proof [13, 25, 37].

Definition 1 (Topology). *A* **Topology** *on set T is a collection $\mathcal{T}$ of subsets of T having the following properties:*

1. $\emptyset \in \mathcal{T}$ *and* $T \in \mathcal{T}$.

2. $\mathcal{T}$ *is closed under arbitrary unions. i.e. For any collection* $\{A_i\}_{i\in I}$*, if for all* $i \in I, A_i \in \mathcal{T}$*, then* $\bigcup\limits_{i\in I} A_i \in \mathcal{T}$.

3. $\mathcal{T}$ *is closed under finitely many unions. i.e. For any finite collection* $\{A_i\}_{i\in I}$*, if for all* $i \in I$*,* $A_i \in \mathcal{T}$*, then* $\bigcap\limits_{i\in I} A_i \in \mathcal{T}$.

Given a set T and a topology $\mathcal{T}$ defined on T, the pair $(T, \mathcal{T})$ is called a **topological space**.

Given a topological space $(T, \mathcal{T})$, all sets $A \in \mathcal{T}$ are called **open sets**.

Note: From this point on, when I say T is a topological space, I assume I are talking about any topology $\mathcal{T}$ on T.

Definition 2 (**σ-algebra**). *Let T be a set. A collection Σ of subsets of T is a* **σ-field** *or a* **σ-algebra** *over T if and only if $T \in \tau$ and τ is closed under countable unions, intersections and complements, i.e.,*

1. *$T \in \Sigma$ and $\emptyset \in \Sigma$.*

2. *$A \in \Sigma$ implies $A^c \in \Sigma$.*

3. *Σ is closed under countable unions, i.e. For any countable collection $\{A_i\}_{i \in I}$, if for all $i \in I$, $A_i \in \Sigma$, then $\bigcap_{i \in I} A_i \in \Sigma$.*

A **measurable space** *is a pair (T, Σ) such that T is a set and Σ is a σ-algebra over T.*

Definition 3 (**Measure**). *$m : \Sigma \to \mathbb{R} \cup \{-\infty, \infty\}$ is a measure over a measurable space (T, Σ) if*

1. *$m(A) \geq m(\emptyset) = 0$ for all $A \in \Sigma$,*

2. *For all countable collection $\{A_i\}_{i \in I}$ of pairwise disjoint sets in Σ,*

$$m(\bigcup_{i \in I} A_i) = \sum_{i \in I} m(A_i)$$

Given a measurable space (T, Σ), a measure π on (T, Σ) is called **probability measure** *if $\pi(T) = 1$.*

I call the tuple (T, Σ, π) a **probability space**, *if π is a probability measure over the measurable space (T, Σ).*

Given a set T, a collection of subsets $A_\alpha \subseteq T$ (not necessarily countable), I denote the smallest σ-algebra Σ such that $A_\alpha \in \Sigma$ for all α by $\sigma(\{A_\alpha\})$.

Definition 4 (Borel σ-algebra). *Given a topological space T, a Borel σ-algebra, $\mathcal{B}(T)$ is the smallest σ-algebra containing all open sets of T.*

$(T, \mathcal{B}(T))$ is called a **Borel Space** *when T is a topological space and $\mathcal{B}(T)$ is a Borel σ-algebra over T.*

Consider a topological space $(T, \mathcal{T})$, for any set $A \in \mathcal{T}$, $(A, \mathcal{T}_A)$ is also a topological space (where $\mathcal{T}_A = \{E \cap A | E \in \mathcal{T}\}$).

Given a measurable space (T, Σ), for any set $A \in \Sigma$, (A, Σ_A) is also a measurable space (where $\Sigma_A = \{E \cap A | E \in \Sigma\}$).

Topology and σ-algebra over Reals: Consider the smallest topology R over the real space $\mathbb{R}$ which contains all intervals $(a, \infty) \subseteq \mathbb{R}$ for all $-\infty < a < \infty$. To avoid confusion, I will refer the topological space over $\mathbb{R}$ with the symbol $\mathcal{R}$. I can now use this topology to define a borel σ-algebra $\mathcal{B}(\mathcal{R})$ over this topological space. Using the above defined topology and σ-algebra I can define a topological space and a σ-algebra for any open or closed intervel in $\mathbb{R}$.

Definition 5 (Measurable Function over Measurable Spaces). *Given measurable spaces (T_1, Σ_1) and (T_2, Σ_2), a function $h : T_1 \to T_2$ is a measurable function from (T_1, Σ_1) to (T_2, Σ_2) if $h^{-1}\{B\} \in \Sigma_1$ for all sets $B \in \Sigma_2$, where $h^{-1}\{B\} = \{x : h(x) \in B\}$. [13]*

The measurable function h is also know as a **Random Variable** *from measurable space (T_1, Σ_1) to (T_2, Σ_2).*

If $h : T_1 \to T_2$ is a measurable function from measurable space (T_1, Σ_1) to a measurable space (T_2, Σ_2), and let π be a probability measure on (T_1, Σ_1), then $\pi_h : \Sigma_2 \to [0, 1]$ defined as

$$\pi_h(A) = \pi(h^{-1}(A))$$

is a probability measure on (T_2, Σ_2).

Definition 6 (Pushforward measure). *Given a probability space (T_1, Σ_1, π) and a measurable function to a measurable space (T_2, Σ_2), then the pushforward measure of*

π is defined as a probability measure $f_*(\pi) : \Sigma_2 \to [0,1]$ given by

$$(f_*(\pi))(B) = \pi(f^{-1}(B)) \text{ for } B \in \Sigma_2$$

Definition 7 (Measurable). *A function f is measurable if f is a measurable function from a measurable space (T, Σ) to $(\mathbb{R}, \mathcal{B}(\mathcal{R}))$.*

f is measurable if and only if

$$\forall a \in \mathbb{R}.\{x \in T | f(x) > a\} \in \Sigma$$

Intuitively, for every real number $-\infty < a < \infty$, there exists a set $A \in \tau$ containing all elements which f maps to real numbers greater than a.

Definition 8 (Simple Function). *Given a measurable space (T, Σ), $s : T \to [0, \infty)$ is a simple function if $s(t) = \Sigma_{i=1}^{N} a_i I_{A_i}(t)$, where $a_i \in [0, \infty)$, $A_i \in \Sigma$, $I_{A_i}(t) = 1$ if $t \in A_i$ and 0 otherwise, and the A_i are disjoint.*

Definition 9 (Lebesgue Integral). *Given a measurable space (T, Σ) and a measure m over (T, Σ), for each $A \in \Sigma$ and disjoint $A_i \in \Sigma$, I define*

$$\int_A I_{A_i}(t) m(dt) = m(A_i \cap A)$$

Hence

$$\int_A s(t) m(dt) = \sum_{i=1}^{N} a_i m(A_i \cap A)$$

Given a function $f : T \to [0, \infty)$ which is measurable, I define

$$\int_A f(t) m(dt) = \mathsf{sup}\{\int_A s(t) m(dt) | s(t) \text{ is simple }, 0 \le s \le f\}$$

where $s \le f$ if $\forall t.s(t) \le f(t)$, as the Lebesgue Integral of function f over a set E in measurable space (T, τ) with measure m.

Given a function $f : T \to \mathbb{R}$ which is measurable, I define

$$\int_A f(t)m(dt) = \int_A f_+(t)m(dt) - \int_A f_-(t)m(dt)$$

where $f_+(t) = \max(f(t), 0)$ and $f_-(t) = \max(-f(t), 0)$. An integral of a measurable function f is the sum of the integral of the positive part and the integral of the negative part.

Note: *From this point on, $\int f(t)m(dt)$ denotes $\int_T f(t)m(dt)$ where T is the set over which the measurable space and measure m is defined.*

Definition 10 (Markov Transition Kernel). *Let (T, Σ) be a measurable space. A Markov Transition kernel on (T, Σ) is a map $K : T \times \Sigma \to [0, 1]$ such that :*

1. *for any fixed $A \in \Sigma$, the function $K(., A)$ is measurable function from (T, Σ) to $[0, 1]$.*

2. *for any fixed $t \in T$, the function $K(t, .)$ is a probability measure on (T, Σ).*

Definition 11 (π-irreducible). *Given a probability space (T, τ, π), a Markov Transition Kernel $K : T \times \tau \to [0, 1]$ is π-irreducible if for each $t \in T$ and each $A \in \tau$, such that $\pi(A) > 0$, there exists an integer $n = n(t, A) \geq 1$ such that*

$$K^n(t, A) > 0$$

where $K^n(t, A) = \int_T K^{n-1}(t, dt')K(t', A)$ and $K^1(t, A) = K(t, A)$.

Definition 12 (Stationary Distribution). *Given a probability space (T, τ, π), π is the stationary distribution of a π-irreducible Markov Transition Kernel $K : T \times \tau \to [0, 1]$ if*

$$\pi K = \pi$$

where $(\pi K)(A) = \int K(t, A)\pi(dt)$.

Definition 13 (Aperiodicity). *Given a probability space (T, τ, π), a π-irreducible Markov Transition Kernel $K : T \times \tau \to [0, 1]$ is periodic if there exists an integer*

$d \geq 2$ and a sequence $\{E_0, E_1, \ldots E_{d-1}\}$ and N of d non-empty disjoint sets in τ such that, for all $i = 0, 1, \ldots d-1$ and for all $t \in E_i$,

1. $(\cup_{i=0}^{d} E_i) \cup N = T$
2. $K(t, E_j) = 1$ *for* $j = i + 1(\mathsf{mod}\ d)$
3. $\pi(N) = 0$

Otherwise K is aperiodic.

Definition 14 (Asymptotic convergence). *Given a probability space (T, τ, π) and sample $t \in T$, a Markov Transition Kernel $K : T \times \tau \rightarrow [0,1]$ is said to asymptotically converge to π if*

$$\mathsf{lim}_{n \to \infty} ||K^n(t,.) - \pi|| = 0$$

where $||.||$ refers to the total variation norm of a measure λ, defined over measurable space (T, τ), defined as

$$||\lambda|| = \mathsf{sup}_{A \in \tau} \lambda(A) - \mathsf{inf}_{A \in \tau} \lambda(A)$$

Theorem 3. *Given a probability space (T, τ, π) and a Markov Transition Kernel $K : T \times \tau \rightarrow [0,1]$. If K is π-irreducible, aperiodic, and $\pi K = \pi$ holds, then for π-almost all t,*

$$\mathsf{lim}_{n \to \infty} ||K^n(t,.) - \pi|| = 0$$

i.e., K converges to π. π-almost all t means that there exists a set $D \subseteq T$ such that $\pi(D) = 1$ and for all $t \in D$, $\mathsf{lim}_{n \to \infty} ||K^n(t,.) - \pi|| = 0$.

Athreya, Doss, and Sethuraman (1992) present the proof for this theorem [2].

All popular asymptotically converging Markov Chain Algorithms, like variants of the Metropolis Hasting and Gibbs Algorithm, when parameterized over probability space (T, τ, π), are π-irreducible and aperiodic with stationary distribution π.

Definition 15 (Subalgebra). *$\mathcal{E}$ is a σ-subalgebra of a measurable space (T, τ) if $\mathcal{E}$ is a σ-algebra of some set $A \subseteq T$ and $\mathcal{E} \subseteq \tau$.*

Definition 16 (Induced Probability space). *Given a probability space* (T, τ, π) *and* $A \in \tau$, *define* $\tau_A = \{B \cap A | B \in \tau\}$. *Note that since* τ *is a* σ*-algebra,* τ_A *is a sub-algebra over set* A. (A, τ_A) *is a measurable space. If* $\pi(A) > 0$, *then function* $\pi_A : \tau_A \to [0, 1]$, *defined as* $\pi_A(x) = \pi(x)/\pi(A)$, *is a probability measure over* (A, τ_A). (A, τ_A, π_A) *is the probability space induced by* $A \in \tau$.

4.2 Class functions and Class Kernels

I next introduce the concept of class functions and class kernels which formalize the concept of subproblem based inference metaprograms. To aid the reader, I reduce the concepts to analyze the special case of Gibbs sampling.

Definition 17 (Two way measurable function). *Given a measurable space* (T_1, Σ_1) *and a measurable space* (T_2, Σ_2), *a measurable function* f *from* (T_1, Σ_1) *to* (T_2, Σ_2) *is a* **two way measurable function**, *if for all sets* $A \in \Sigma_1$ *there exists a set* $B \in \Sigma_2$, *such that*

$$B = \{f(t) | t \in A\}$$

i.e., the map of any set $A \in \Sigma_1$ *is a set in* Σ_2.

Given a two way measurable function f, *I can trivially extend* f *to a function* $g : \Sigma_1 \to \Sigma_2$ *between sets in* Σ_1 *to sets in* Σ_2, *where* $g(A) = \{f(t) | t \in A\}$. *Since* f *is a measurable function, the function* $f^{-1} : \Sigma_2 \to \Sigma_1$ *is also defined which maps sets in* Σ_2 *to sets in* Σ_1.

Note: *From this point on, given a two way measurable function* f, *I will simply use it to represent function* g *defined above, mapping sets from* Σ_1 *to* Σ_2. *I also use* f^{-1} *as a reverse map from* Σ_2 *to* Σ_1 *defined above. Note that* f^{-1} *may or may not be the inverse of the function* f.

Example 1. *Given measurable spaces* $(X, \mathcal{X})$, $(Y, \mathcal{Y})$ *and product space* $(X \times Y, \mathcal{X} \otimes \mathcal{Y})$, *the projection functions* proj_x *and* proj_y *are two way measurable functions from* $(X \times Y, \mathcal{X} \otimes \mathcal{Y})$ *to* $(X, \mathcal{X})$ *and* $(Y, \mathcal{Y})$, *respectively.*

Definition 18 (Generalized Product Space). *Given countable sets of disjoint measurable spaces* $\{(X_i, \mathcal{X}_i)|i \in I\}$ *and* $\{(Y_i, \mathcal{Y}_i)|i \in I\}$*, I construct a countable set of product spaces*

$$\{(X_i \times Y_i, \mathcal{X}_i \otimes \mathcal{Y}_i)|i \in I\}$$

Given the product spaces defined above, I define **generalized product space** $(C, \mathcal{C})$ *as a union of product spaces,*

$$(C, \mathcal{C}) = (\bigcup_{i \in I} X_i \times Y_i, \sigma(\bigcup_{i \in I} \mathcal{X}_i \otimes \mathcal{Y}_i))$$

Given a probability measure π *on a generalized product space* $(C, \mathcal{C})$*, I can define a conditional distribution* π_i *over each product space* $(X_i \times Y_i, \mathcal{X}_i \otimes \mathcal{Y}_i)$ *such that*

$$\pi(A) = \sum_{i \in I} \pi_i(A \cap X_i \times Y_i)\pi(X_i \times Y_i)$$

I assume, for each product probability space $(X_i \times Y_i, \mathcal{X}_i \otimes \mathcal{Y}_i, \pi_i)$ *I can construct a regular conditional probability measure* $v_{\pi_i} : X_i \times \mathcal{Y}_i \to [0,1]$*, such that for each* $U \times V \in \mathcal{X}_i \otimes \mathcal{Y}_i$

$$\pi_i(U \times V) = \int_U v_{\pi_i}(x, V)\pi_i(dx \times Y_i)$$

Note: *When constructing the generalized product space, one has to prove the above assumption i.e., the regular conditional probability measure exists.*

Example 2. *Given measurable spaces* $(X, \mathcal{X})$, $(Y, \mathcal{Y})$*, the product space* $(X \times Y, \mathcal{X} \otimes \mathcal{Y})$ *is a* **generalized product space** *if and only if, given a probability measure* π *on the product space, the regular conditional probability* $v_\pi : X \times \mathcal{Y} \to [0,1]$ *is defined.*

Definition 19 (Class functions). *Given a measurable space* (T, Σ) *and a generalized product space* $(C, \mathcal{C})$*, a* **Class function** *is a one-to-one two way measurable function* f *from* (T, Σ) *to* $(C, \mathcal{C})$*.*

Given a class function f *and the target product space* $(C, \mathcal{C})$*, the* **projection**

functions

$$f_x = \mathsf{proj}_x \circ f : T \to \bigcup_{i \in I} X_i \textit{ and } f_y = \mathsf{proj}_y \circ f : T \to \bigcup_{i \in I} Y_i$$

are also a two way measurable functions from space (T, Σ) *to projection spaces* $\big(\bigcup_{i\in I} X_i, \sigma(\bigcup_{i\in I} \mathcal{X}_i)\big)$ *and* $\big(\bigcup_{i\in I} Y_i, \sigma(\bigcup_{i\in I} \mathcal{Y}_i)\big)$ *respectively.*

Example 3. *Given measurable space* $(X, \mathcal{X})$, $(Y, \mathcal{Y})$ *and generalized product spaces* $(X \times Y, \mathcal{X} \otimes \mathcal{Y})$ *and* $(Y \times X, \mathcal{Y} \otimes \mathcal{X})$, *the identity function* $id : X \times Y \to X \times Y$ *and the reverse function* $re : X \times Y \to Y \times X$ *(i.e.* $re(\langle x, y\rangle) = \langle y, x\rangle$*) are class functions from* $(X \times Y, \mathcal{X} \otimes \mathcal{Y})$ *to* $(X \times Y, \mathcal{X} \otimes \mathcal{Y})$ *and* $(Y \times X, \mathcal{Y} \otimes \mathcal{X})$.

Note that $id_x = \mathsf{proj}_x$ *and* $re_x = \mathsf{proj}_y$.

Definition 20 (Connecting the space (T, Σ, π)). *Given a probability space* (T, Σ, π), *a finite set of class functions* $\mathcal{F} = \{f_1, f_2 \dots f_n\}$ *connect the probability space* (T, Σ, π), *if for all sets* $A \in \Sigma$ *and any two functions* $f, g \in \mathcal{F}$

$$\begin{aligned} \pi(f_x^{-1}(f_x(A)) \cap g_x^{-1}(g_x(A)^c)) = \pi(f_x^{-1}(f_x(A)^c) \cap g_x^{-1}(g_x(A))) = 0 \\ \implies \pi(f_x^{-1}(f_x(A)^c)) = 0 \textit{ or } \pi(g_x^{-1}(g_x(A)^c)) = 0 \end{aligned}$$

Example 4 (Connected product space). *Given a probability space* $(X \times Y, \mathcal{X} \otimes \mathcal{Y}, \pi)$ *and class functions id and re, then id and re connect the space* $(X \times Y, \mathcal{X} \otimes \mathcal{Y}, \pi)$, *if for all sets* $U \times V \in \mathcal{X} \otimes \mathcal{Y}$,

$$\pi(U \times V^c) = \pi(U \times V^c) = 0 \implies \pi(U^c \times Y) = 0 \textit{ or } \pi(X \times V^c) = 0$$

as

$$\pi(id_x^{-1}(id_x(U \times V)) \cap re_x^{-1}(re_x^{-1}(U \times V)^c) = \pi(U \times Y \cap X \times V^c) = \pi(U \times V^c)$$

and

$$\pi(id_x^{-1}(id_x(U \times V)^c) \cap re_x^{-1}(re_x^{-1}(U \times V)) = \pi(U^c \times Y \cap X \times V) = \pi(U^c \times V)$$

Definition 21 (Class Kernels). *Given a probability space (T, Σ, π), a class function f to a generalized product space $(C^f, \mathcal{C}^f) = \big(\bigcup_{i \in I} X_i^f \times Y_i^f, \sigma(\bigcup_{i \in I} \mathcal{X}_i^f \otimes \mathcal{Y}_i^f)\big)$, f maps each point $t \in T$ to some point $\langle x, y \rangle$ in $X_i^f \times Y_i^f$ for some i.*

For each $i \in I$, I define a function $K_i : X_i^f \to (Y_i^f \times \mathcal{Y}_i^f) \to [0, 1]$ such that

- *$K_i(x) : Y_i^f \times \mathcal{Y}_i^f \to [0, 1]$ is a $v_{f_*(\pi)_i}(x, .)$-irreducible, aperiodic Markov Transition Kernel with $v_{f_*(\pi)_i}(x, .)$ as it's stationary distribution.*

- *$K_i(.)(y, A)$ is measurable for all $y \in Y_i^f$ and all $A \in \mathcal{Y}_i^f$.*

Given these kernel functions, for each i I define a **Class Kernel** *$K_f : T \times \Sigma \to [0, 1]$ as a Markov Transition Kernel defined as*

$$K_f(t, A) = K_i(x)(y, V) I(x, U)$$

where $f(t) = \langle x, y \rangle \in X_i \times Y_i$ and $U \times V = f(A) \cap (X_i \times Y_i)$.

4.3 Properties of Class kernels

Consider a probability space (T, Σ, π) and a class function f to a generalized product space $(C^f, \mathcal{C}^f) = \big(\bigcup_{i \in I} X_i^f \times Y_i^f, \sigma(\bigcup_{i \in I} \mathcal{X}_i^f \otimes \mathcal{Y}_i^f)\big)$. Below, I prove properties of a Class Kernel K_f within this context.

Lemma 8. *For all $t \in T$ and $A \in \Sigma$,*

$$K_f^n(t, A) = K_i^n(x)(y, V) I(x, U)$$

where $f(t) = \langle x, y \rangle \in X_i \times Y_i$ and $U \times V = f(A) \cap X_i \times Y_i$.

Proof. Proof by induction.

Base case:

$$K_f(t, A) = K_i(x)(y, V) I(x, U)$$

using definition of K_f.

Induction Hypothesis:

For all $1 \leq n \leq m$, the following statement is true

$$K_f^n(t, A) = K_i^n(x)(y, V)I(x, U)$$

Induction Case:

$$K_f^{m+1}(t, A) = \int_{t' \in T} K_f^m(t', A)K_f(t, dt')$$

$$= \int_{x' \in X_i} \int_{y' \in Y_i} K_f^m(f(x', y'), A)K_i(x)(y, dy')I(x, dx')$$

Note that $f(x', y') \in X_i \times Y_i$

$$= \int_{x' \in X_i} \int_{y' \in Y_i} K_i^m(x')(y', V)I(x', U)K_i(x)(y, dy')I(x, dx')$$

$$= \int_{y' \in Y_i} K_i^m(x)(y', V)I(x, U)K_i(x)(y, dy')$$

$$= \int_{y' \in Y_i} K_i^m(x)(y', V)I(x, U)K_i(x)(y, dy')$$

$$= K_i^{m+1}(x)(y, V)I(x, U)$$

Hence proved. □

Lemma 9.

$$\pi(A) = \int_{t \in T} K_f(t, A)\pi(dt)$$

Proof. Every set $A \in \Sigma$ can be written as $f^{-1}(U \times V)$ for some $U \times V \in \mathcal{C}^f$ as f is a two-way measurable function.

$$\int_{t \in T} K_f(t, A)\pi(dt) = \int_{t \in T} K_f(t, f^{-1}(U \times V))\pi(f^{-1}(dt))$$

I can split the integral into sum over the constituent product spaces $X_i \times Y_i$,

$$= \sum_{i \in I} \int_{x \in X_i} \int_{y \in Y_i} K_f(f^{-1}(x, y), f^{-1}(U \times V))f_*(\pi)_i(dx \times dy)f_*(\pi)(X_i \times Y_i)$$

I can rewrie $K_f(f^{-1}(x,y), f^{-1}(U \times V))$ as $K(x)(y, V \cap Y_i)I(x, U \cap X_i)$.

$$= \sum_{i \in I} \int_{x \in X_i} \int_{y \in Y_i} K(x)(y, V \cap Y_i)I(x, U \cap X_i) f_*(\pi)_i(dx \times dy) f_*(\pi)(X_i \times Y_i)$$

$\int_{x \in X_i} f(x)I(x, U \cap X_i)m(dx) = \int_{x \in U \cap X_i} f(x)m(dx)$ as $I(x, U \cap X_i)$ is zero for all $x \notin U \cap X_i$.

$$= \sum_{i \in I} \int_{y \in Y_i} \int_{x \in U \cap X_i} K(x)(y, V \cap Y_i) f_*(\pi)_i(dx \times dy) f_*(\pi)(X_i \times Y_i)$$

I can rewrite $f_*(\pi)_i(dx' \times dy)$ as $v_{f_*(\pi)_i)}(x, dy) f_*(\pi)_i(dx' \times Y_i)$ using the definition of regular conditional probability distribution.

$$= \sum_{i \in I} \int_{y \in Y_i} \int_{x \in U \cap X_i} K(x)(y, V \cap Y_i) v(x, dy) f_*(\pi)_i(dx \times Y_i) f_*(\pi)(X_i \times Y_i)$$

$$= \sum_{i \in I} \int_{x \in U \cap X_i} \Big(\int_{y \in Y_i} K(x)(y, V \cap Y_i)) v_{f_*(\pi)_i}(x, dy) \ \Big) f_*(\pi)_i(dx \times Y_i) f_*(\pi)(X_i \times Y_i)$$

$\int_{y \in Y_i} K_i(x)(y, V \cap Y_i) v_{f_*(\pi)_i}(x, dy) = v_{f_*(\pi)_i}(x, V \cap Y_i)$ as $v_{f_*(\pi)_i}(x, .)$ is the stationary distribution for kernel $K_i(x)$.

$$= \sum_{i \in I} \int_{x \in U \cap X_i} v_{f_*(\pi)_i}(x, V \times Y_i) f_*(\pi)_i(dx \times Y_i) f_*(\pi)(X_i \times Y_i)$$

$$= \sum_{i \in I} f_*(\pi)_i(U \times V \cap X_i \times Y_i) f_*(\pi)(X_i \times Y_i) = f_*(\pi)(U \times V) = \pi(A)$$

□

Lemma 10. *K_f is aperiodic if for at least one $x \in X_i$ for some $i \in I$, $K_i(x) : Y_i \times \mathcal{Y}_i \to [0,1]$ is aperiodic.*

Proof. Proof by contradiction. Let me assume K_f is periodic, i.e., there exists an integer $d \geq 2$, and a sequence $\{E_0, E_1, \ldots E_{d-1}\}$ and N of d non-empty disjoint sets in τ such that, for all $i = 0, 1, \ldots d-1$ and for all $t \in E_i$,

1. $(\cup_{i=0}^{d} E_i) \cup N = T$

2. $K_f(t, E_j) = 1$ for $j = i + 1 (\mathsf{mod}\ d)$

3. $\pi(N) = 0$

For all $t \in E_i$, $K_f(t, E_j) = 1$ for $j = i + 1(\mathsf{mod}\ d)$ Consider any $k \in I$, any $x \in X_k$ and $U_i \times V_i = f(E_i) \cap X_k \times Y_k$.

Consider a trace $t \in E_i$, such that $f_x(t) = x$. Since $K_f(t, E_{i+1}) = 1 \leq I(x, U_{i+1})$, for all $i = 0, 1 \ldots d-1$, there exists a trace $t' \in E_i$, such that $f(t') = \langle x, y \rangle$.

$K_f(t, E_{i+1}) = 1 \leq K(x)(y, V_{i+1})$, Hence if K_f is aperiodic, then for all x, $K(x)$ is aperiodic.

□

4.4 Stochastic Alternating Class Kernels

Consider a probability space (T, Σ, π) where Σ is countably generated. I construct a new Markov Chain Transition Kernel using a finite set of class functions $\mathcal{F} = \{f_1, f_2 \ldots f_m\}$ and respective class Kernels $K_{f_1}, K_{f_2} \ldots K_{f_m}$.

Given m positive real numbers $p_k \in (0, 1)$ which sum to 1 (i.e. $\sum\limits_{k=1}^{m} p_k = 1$), I define a stochastic alternating Markov Chain Transition Kernel $K : T \times \Sigma \to [0, 1]$ as

$$K(t, A) = \sum_{k=1}^{m} p_k K_{f_k}(t, A)$$

This transition kernel corresponds to randomly picking a class kernel K_{f_k} with probability p_k and using it to transition into the next Markov Chain State.

I will now consider the question of convergence of this transition kernel.

Let $R_t^k = \{A \in \Sigma \wedge \exists 1 \leq n \leq k. K^n(t, A) > 0\}$ be the set of all sets in Σ which are reachable by kernel K in k steps starting from element $t \in T$.

Consider the limiting case R_t^∞. Let $B_t^\infty = \{t \in T | \forall\ A \in \Sigma. A \notin R_t^\infty \implies t \notin A\}$ which is the set of elements $t' \in T$ such that any set A in Σ which contains t' is reachable by kernel K.

Lemma 11. *For any $f \in \mathcal{F}$, any element $t' \in B_t^\infty$ such that $f(t') = \langle x, y \rangle \in X_i^f \times Y_i^f$,* *and any set $U \times V \in \sigma(\mathcal{X}_i^f \otimes \mathcal{Y}_i^f)$ such that $A = f^{-1}(U \times V)$, the following condition holds true*

$$v_{f_*(\pi)_i}(x, V) > 0 \wedge x \in U \implies A \in R_t^\infty$$

Proof. Consider class Kernel $K_f(t', A) = K_i(x)(y, V)I(x, U)$. Since $v_{f_*(\pi)_i}(x, V) > 0$ and K_i is $v_{f_*(\pi)_i}$-irreducible, there exists an n such that

$$K_i^n(x)(y, V) > 0$$

Since $x \in U$, using Lemma 8

$$K_f^n(t', A) = K_i^n(x)(y, V)I(x, U) > 0$$

$$K_f^n(t', A) > 0 \implies K^n(t', A) > 0$$

Since $t' \in B_t^\infty$, there exists an n' such that, for all sets $B \in \Sigma$ with $t' \in B$, $K^{n'}(t, B) > 0$. Hence

$$K^{n+n'}(t, A) \geq \int_B K^n(t', A)K^{n'}(t, dt') > 0 \implies A \in R_t^\infty$$

□

Lemma 12. *For any positive probability set A and any function $f \in \mathcal{F}$, if $A \subseteq f_x^{-1}(f_x(B_t^\infty))$ then $A \in R_t^\infty$.*

Proof. Given a set A, I can treat $f(A)$ as a union of sets $\{U_i \times V_i | i \in I^f\}$, where $U_i \times V_i$ are elements of set $f(A)$ which are elements of the set $X_i^f \times Y_i^f$ (i.e. $U_i \times V_i = f(A) \cap X_i^f \times Y_i^f$). Since $\pi(A) > 0$, $f_*(\pi)(f(A)) > 0$ and for at least for one $i \in I_f$, $f_*(\pi)_i(U_i \times V_i) > 0$.

Since $A \subseteq f_x^{-1}(f_x(B_t^\infty))$, for each $x \in U_i$ there exists at least one element $t' \in B_t^\infty$ such that $f_x(t') = x$.

If $f_*(\pi)_i(U_i \times V_i) > 0$, there exists at least one $t' \in B_t^\infty$ such that $f(t') = \langle x, y \rangle \in U_i \times Y_i^f$ and $v_{f_*(\pi)_i}(x, V_i) > 0$. Hence $f^{-1}(U_i \times V_i) \in R_t^\infty$. □

Lemma 13. *If $\mathcal{F}$ connects the space (T, Σ, π) then there does not exist a positive probability set $A \in \Sigma$, such that $A \subseteq \bigcap_{f \in \mathcal{F}} f_x^{-1}((f_x(B_t^\infty))^c)$.*

Proof. Proof by Contradiction.

Let me assume such a set A exists. If A is a positive probability set, then $\pi(\bigcap_{f\in\mathcal{F}} f_x^{-1}((f_x(B_t^\infty))^c)) > 0$.

For any two functions $f, g \in \mathcal{F}$,

$$\pi(f_x^{-1}((f_x(B_t^\infty))^c) \cap g_x^{-1}(g_x(B_t^\infty))) = 0$$

The set $f_x^{-1}((f_x(B_t^\infty))^c)$ only contains elements which are not in B_t^∞ and $g_x^{-1}(g_x(B_t^\infty))$ contains elements t' such that there exists at least one element $t'' \in B_t^\infty$ with $g_x(t') = g_x(t'')$.

Any positive probability set $B \subseteq g_x^{-1}(g_x(B_t^\infty))$ is also a subset of B_t^∞ (using Lemma 12). Hence

$$\pi(f_x^{-1}((f_x(B_t^\infty))^c) \cap g_x^{-1}(g_x(B_t^\infty))) = 0$$

Similarly

$$\pi(f_x^{-1}(f_x(B_t^\infty)) \cap g_x^{-1}((g_x(B_t^\infty))^c)) = 0$$

Since $\pi(\bigcap_{f\in\mathcal{F}} f_x^{-1}((f_x(B_t^\infty))^c)) > 0$,

$$\pi(f_x^{-1}(f_x^{-1}((f_x(B_t^\infty))^c)) > 0 \text{ and } \pi(g_x^{-1}((g_x(B_t^\infty))^c)) > 0$$

But this contradicts the fact the $\mathcal{F}$ connects the space (T, Σ, π). Hence no such set A exists. □

Theorem 4. *If $\mathcal{F}$ connects the space (T, Σ, π) then the Markov Transition Kernel K is π-irreducible.*

Proof. Proof by contradiction. Let me assume K is not π-irreducible, then there exists a positive probability set $A \in \Sigma$ such that $A \notin R_t^\infty$, then

If $\pi(A \cap B_t^\infty) > 0$, there exists a set $B \subseteq B_t^\infty$ and $B \subseteq A$ which implies $A \in R_t^\infty$. Hence $\pi(A \cap B_t^\infty) = 0$.

For any $f \in \mathcal{F}$, $\pi(A \cap f_x^{-1}(f_x(B_t^{\infty}))) > 0$, there exists a set $B \in R_t^{\infty}$ and $B \subseteq A$ which implies $A \in R_t^{\infty}$. Hence $\pi(A \cap f_x^{-1}(f_x(B_t^{\infty}))) = 0$.

Since f_x is a 2-way measurable function (and one-one function from sets to sets), for any set B $f_x^{-1}(f_x(B))^c = f_x^{-1}(f_x(B)^c)$.

Since $\pi(A) > 0$ and For any $f \in \mathcal{F}$ $\pi(A \cap f_x^{-1}(f_x(B_t^{\infty}))) = 0$ this means

$$\pi(A \cap \bigcap_{f \in \mathcal{F}} f_x^{-1}(f_x(B_t^{\infty})^c)) > 0$$

which means there exists a positive probability set $B \in \Sigma$ and $B \subseteq f_x^{-1}(f_x(B_t^{\infty})^c))$, which is impossible.

Hence no such set A exists. K is π-irreducible. $\square$

Theorem 5. *π is the stationary distribution of Markov Kernel K, i.e.*

$$\int K(t, A)\pi(dt) = \pi(A) \textit{ for all } A \in \Sigma$$

Proof.

$$\int_{t \in T} K(t, A)\pi(dt) = \int_{t \in T} \sum_{i=1}^{m} p_i K_{f_i}(t, A)\pi(dt)$$

$$= \sum_{i=1}^{m} p_i \int_{t \in T} K_{f_i}(t, A)\pi(dt) = \sum_{i=1}^{m} p_i \pi(A) = \pi(A)$$

$\square$

Theorem 6. *The Markov Transition Kernel K is aperiodic if at least one of the class kernels K_{f_j} is aperiodic.*

Proof. Proof by Contradiction.

Let me assume K is periodic. i.e., there exists an integer $d \geq 2$ and a sequence $\{E_0, E_1, \ldots E_{d-1}\}$ and N of d non-empty disjoint sets in τ such that, for all $i = 0, 1, \ldots d-1$ and for all $t \in E_i$,

1. $(\cup_{i=0}^{d} E_i) \cup N = T$

2. $K(t, E_j) = 1$ for $j = i + 1 (\mathsf{mod}\ d)$

3. $\pi(N) = 0$

If $K(t, E_j) = 1$ then for all $f \in \mathcal{F}$, $K_f(t, E_j) = 1$. Therefore, for all $i = 0, 1, \ldots d-1$ and for all $t \in E_i$, $K_f(t, E_j) = 1$ for $j = i + 1 (\mathsf{mod}\ d)$.

Hence if K is periodic, then for all $f \in \mathcal{F}$, K_f is periodic.

Hence by contradiction, K is aperiodic. □

Theorem 7. *Markov Transition Kernel K converges to probability distribution π.*

Proof. Using Theorems 3, 4, 5, and 6. □

Chapter 5

Inference Metaprograms

Within this Chapter, I formalize the concept of the probability of a trace, introduce inference metaprogramming and use the results in Chapter 4 to prove the convergence of inference metaprograms.

5.1 Preliminaries

Within the Section, I will tie mathematical concepts introduced in Chapter 4 to more concrete concepts used in probabilistic programming.

5.1.1 Probability of a Trace

Given two traces are equal if and only if they differ in the choice of unique identifiers, within this Section, for clarity, I drop the *id*'s associated with augmented expressions and stochastic choices in traces and augmented expressions wherever it is not required.

Assuming a countable set of variable names allowed in my probabilistic lambda calculus language, a countable number of expressions and a countable number of programs can be described in my probabilistic lambda calculus language.

Within my probabilistic lambda calculus language, I assume that all stochastic distributions $\mathsf{Dist}_k : V \times \mathcal{P}(E) \to [0, 1]$ are functions from a tuple of value and set of lambda expressions in my language to a real number between 0 and 1, such that

$$\begin{array}{rcl}
\mathsf{pdf}[\![x : x]\!] & = & 1 \\
\mathsf{pdf}[\![x : v]\!] & = & 1 \\
\mathsf{pdf}[\![\lambda.x\ e : v]\!] & = & 1 \\
\mathsf{pdf}[\![(ae_1\ ae_2)\perp : v]\!] & = & \mathsf{pdf}[\![ae_1]\!] * \mathsf{pdf}[\![ae_2]\!] \\
\mathsf{pdf}[\![(ae_1\ ae_2)x = ae : v]\!] & = & \mathsf{pdf}[\![ae_1]\!] * \mathsf{pdf}[\![ae_2]\!] * \mathsf{pdf}[\![ae]\!] \\
\mathsf{pdf}[\![\mathsf{Dist_i}(ae) = ae_e : v_e]\!] & = & \mathsf{pdf}[\![ae]\!] * \mathsf{pdf}[\![ae_e]\!] * \mathsf{pdf}_{\mathsf{Dist_i}}(\mathcal{V}(ae), e) \\
& \text{where} & p = ae_e \Rightarrow_r e \\
\\
\mathsf{pdf}[\![\emptyset]\!] & = & 1 \\
\mathsf{pdf}[\![\mathsf{assume}\ x = ae; t]\!] & = & \mathsf{pdf}[\![ae]\!] * \mathsf{pdf}[\![t]\!] \\
\mathsf{pdf}[\![\mathsf{observe}(\mathsf{Dist}(ae) = e); t]\!] & = & \mathsf{pdf}_{\mathsf{Dist}}(\mathcal{V}(ae), e) * \mathsf{pdf}[\![ae]\!] * \mathsf{pdf}[\![t]\!]
\end{array}$$

Figure 5-1: Probabilistic measure over traces

for any $v \in V$, $\mathsf{Dist}_k(v, .)$ is a probability measure over probability space $(E, \mathcal{P}(E))$. I assume that for each distribution Dist_k, I am given a probability density function $\mathsf{pdf}_{\mathsf{Dist}_k} : V \times E \to [0, 1]$, such that

$$\mathsf{Dist}_k(v, A) = \sum_{e \in A} \mathsf{pdf}_{\mathsf{Dist}_k}(v, e)$$

Let $T_p = \mathsf{Traces}(p)$ be the set of valid traces of a program p. T_p contains a countable number of traces. I define a σ-algebra Σ_p over set T_p such that, for all $t \in T_p$, $\{t\} \in \Sigma_p$. Given a trace t, $\mathsf{pdf}[\![t]\!]$ is the unnormalized probability density of the trace t (Figure 5-1). The normalized probability distribution $\mu_p : \Sigma_p \to [0, 1]$ for a given program p is defined as

$$\mu_p(A) = \frac{\sum\limits_{t \in A} \mathsf{pdf}[\![t]\!]}{\sum\limits_{t \in T_p} \mathsf{pdf}[\![t]\!]}$$

5.1.2 Reversible Subproblem Selection Strategy

Let p be a probabilistic program, t and t' be valid traces from program p (i.e. $t, t' \in \mathsf{Traces}(p)$), and SS be a subproblem selection strategy that returns a valid subproblem over t.

Definition 22 (Reversible subproblem selection strategy). *A subproblem se-*

lection strategy SS *is reversible if for any two traces* $t, t' \in \mathsf{Traces}(p)$,

$$\mathsf{SS}(t) \vdash t \equiv t' \iff \mathsf{SS}(t') \vdash t' \equiv t$$

i.e., if given a trace t and a valid subproblem $SS(t)$ on trace t, an inference algorithm can modify trace t to achieve trace t', then given trace t' and a valid subproblem $SS(t')$ on t', an inference algorithm can modify trace t' to achieve trace t.

In essence, reversible subproblem selection strategies always allow a subproblem based inference algorithm to reverse the changes it has made to a trace.

I use the shorthand $\mathsf{SS} \vdash t \equiv t'$ to denote $\mathsf{SS}(t) \vdash t \equiv t'$. Note that $\mathsf{SS} \vdash t \equiv t' \iff \mathsf{SS} \vdash t' \equiv t$ for reversible subproblem selection strategies.

Theorem 8. *A reversible subproblem selection strategy* SS *divides the trace space of program p into equivalence classes.*

Proof. $\mathsf{SS} \vdash t \equiv t'$ is an equivalence relation over traces $t, t' \in T$.

Reflexivity : $\mathsf{SS} \vdash t \equiv t$ is true by definition.

Symmetry : By definition of reversibility.

Transitivity : $\mathsf{SS} \vdash t_1 \equiv t_2$, $\mathsf{SS} \vdash t_2 \equiv t_3$ then $\mathsf{SS} \vdash t_1 \equiv t_3$ (by definition of $\equiv$). □

A reversible subproblem selection strategy SS divides the trace space T_p into a countable number of equivalence classes where each equivalence class contains traces which can be modified into any other trace in that class under the subproblem selection strategy.

A trace from a equivalence class cannot be modified by any subproblem based inference algorithm to a trace from a different equivalence class under the given subproblem selection strategy.

Given a reversible subproblem selection strategy SS, let $\mathcal{C}_{SS} = \{c_1, \ldots c_n, \ldots\}$ be the countable set of equivalence classes created by SS over the trace space T_p and $\{T_{c_1}, \ldots T_{c_n}, \ldots\}$ be the equivalence partitions created by SS over T_p.

Note that for all $c \in \mathcal{C}_{SS}$ and $t, t' \in T_c$,

$$\mathsf{SS} \vdash t \equiv t' \;\wedge\; \mathsf{SS} \vdash t' \equiv t$$

and for all $c_i, c_j \in \mathcal{C}_{SS}, t \in T_{c_i}$ and $t' \in T_{c_j}$, where $c_i \neq c_j$

$$(\mathsf{SS} \vdash t \equiv t' \ \vee \ \mathsf{SS} \vdash t' \equiv t) = \mathsf{false}$$

In practice, subproblems are often specified by associating labels with stochastic choices, then specifying the labels whose stochastic choices should be included in the subproblem [24]. A standard strategy is to have a fixed set of labels, with the labels partitioning the choices into classes. Any strategy that always specifies the subproblem via a fixed subset of labels is reversible. Because of this property, all of the subproblem selection strategies presented in [24] are reversible.

Any subproblem selection strategy that always selects a fixed set of variables is also reversible. This property ensures that the subproblem selection strategy in Block Gibbs sampling, for example, is reversible. Hence if two traces only differ in the choice of their id's, all reversible subproblem selection strategy will assign them to the same equivalence class.

5.1.3 Class functions given a subproblem selection strategy

Consider a reversible subproblem selection strategy SS which creates equivalence classes $\mathcal{C}_{SS} = \{c_1, \ldots c_n \ldots\}$ and equivalence partitions $\{T_{c_1}, \ldots T_{c_n}, \ldots\}$. I create a generalized product space and class functions using the given subproblem selection strategy.

Consider the countable set of disjoint measurable spaces

$$\{(C_1, \mathcal{C}_1), \ldots (C_n, \mathcal{C}_n), \ldots\}$$

where $C_k = \{c_k\}$ and $\mathcal{C}_k = \{\emptyset, C_k\}$, and

$$\{(T_{c_1}, \Sigma_{c_1}), \ldots (T_{c_n}, \Sigma_{c_n}), \ldots\}$$

where $\Sigma_{c_k} = \{A \cap T_{c_k} | A \in \Sigma_p\}$, I construct the generalized product space

$$(C, \mathcal{C}) = (\bigcup_{c_i \in \mathcal{C}_{SS}} C_i \times T_{c_i}, \sigma(\bigcup_{c_k \in \mathcal{C}_{SS}} \mathcal{C}_k \otimes \Sigma_{c_k}))$$

Given π a probability measure on $(C, \mathcal{C})$, I can compute the conditional distribution π_i on $(C_i \times T_{c_i}, \mathcal{C}_{SS} \otimes \Sigma_{c_k})$, when $\pi(C_i \times T_{c_i}) > 0$ where

$$\pi_i(A) = \frac{\pi(A)}{\pi(C_i \times T_{c_i})}$$

and then I can trivially define the regular conditional probability measure $v_i : C_i \times \Sigma_{c_i} \to [0, 1]$, where

$$v_i(c_i, A) = \pi_i(C_i \times A)$$

I create the class function $f_{SS} : T_p \to C$, where

$$f_{SS}(t) = \langle c, t \rangle \quad \text{where } c \text{ is the equivalence class of trace } t$$

Since f_{SS} is a one-to-one function, it is trivial to see that f_{SS} is a two way measurable function.

5.1.4 Probability of the subtraces

For all traces $t, t' \in T_p$, such that $\mathsf{SS} \vdash t \equiv t'$, subtraces $t_s = \mathsf{ExtractTraces}(t, \mathsf{SS}(t))$ and $t'_s = \mathsf{ExtractTraces}(t', \mathsf{SS}(t'))$ are traces from the same program, i.e., $t_s, t'_s \in \mathsf{Traces}(p_s)$, where p_s is the subprogram (Soundness).

Similarly, for all traces $t \in T_p$ and subtraces $t_s = \mathsf{ExtractTrace}(t, \mathsf{SS}(t))$, for all subtraces $t'_s \in \mathsf{Traces}(p_s)$ (where $p_s = \mathsf{Program}(t_s)$) and $t' = \mathsf{StitchTrace}(t, t'_s, \mathsf{SS}(t))$ then $\mathsf{SS} \vdash t \equiv t'$ (Completeness).

Hence given a equivalence class c_i and partitioned trace space T_{c_i} created by subprogram selection strategy SS, there exists a subprogram p_s, such that for all traces $t \in T_{c_i}$, subtraces $t_s = \mathsf{ExtractTrace}(t, \mathsf{SS}(t))$ are valid traces of p_s i.e. $t_s \in T_{p_s}$. I can therefore associate traces from an equivalence class to valid subtraces of a subprogram.

Lemma 14. *For any augmented expression ae and subproblem $\mathcal{S}$,*

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s \implies \mathsf{pdf}[\![ae]\!] = \mathsf{pdf}[\![ae_s]\!] * \mathsf{pdf}[\![t_s]\!]$$

Proof. Proof by Induction

Base Case:

Case 1: $ae = x : x$,

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash x : x \Rightarrow_{ex} x : x, \emptyset$$

Then $ae_s = x : x$ and $t_s = \emptyset$.

By definition of pdf, $\mathsf{pdf}[\![x : x]\!] = 1$ and $\mathsf{pdf}[\![\emptyset]\!] = 1$. Therefore

$$\mathsf{pdf}[\![ae]\!] = \mathsf{pdf}[\![ae_s]\!] * \mathsf{pdf}[\![t_s]\!]$$

Case 2: $ae = x : v$,

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash x : v \Rightarrow_{ex} x : v, \emptyset$$

Then $ae_s = x : v$ and $t_s = \emptyset$.

By definition of pdf, $\mathsf{pdf}[\![x : v]\!] = 1$ and $\mathsf{pdf}[\![\emptyset]\!] = 1$. Therefore

$$\mathsf{pdf}[\![ae]\!] = \mathsf{pdf}[\![ae_s]\!] * \mathsf{pdf}[\![t_s]\!]$$

Case 3: $ae = \lambda.x\ e : v$,

By assumption

$$\mathcal{S} \vdash ae \Rrightarrow_{ex} ae_s, t_s$$

By definition of $\Rrightarrow_{ex}$

$$\mathcal{S} \vdash \lambda.x\ e : v \Rrightarrow_{ex} \lambda.x\ e : v, \emptyset$$

Then $\mathsf{pdf}[\![\lambda.x\ e : v]\!] = v, 1$ and $\mathsf{pdf}[\![\emptyset]\!] = 1$. Therefore

$$\mathsf{pdf}[\![ae]\!] = \mathsf{pdf}[\![ae_s]\!] * \mathsf{pdf}[\![t_s]\!]$$

Induction Case:

Case 1: $ae = (ae_1\ ae_2) \perp: v$,

By assumption

$$\mathcal{S} \vdash ae \Rrightarrow_{ex} ae_s, t_s$$

By definition of $\Rrightarrow_{ex}$

$$\mathcal{S} \vdash (ae_1\ ae_2) \perp: v \Rrightarrow_{ex} (ae_s^1\ ae_s^2) \perp: v, t_s^1; t_s^2$$

Then $ae_s = (ae_s^1\ ae_s^2) \perp: v$, $t_s = t_s^1; t_s^2$, $\mathcal{S} \vdash ae_1 \Rrightarrow_{ex} ae_s^1, t_s^1$, and $\mathcal{S} \vdash ae_2 \Rrightarrow_{ex} ae_s^2, t_s^2$.

By induction hypothesis

$$\mathcal{S} \vdash ae_1 \Rrightarrow_{ex} ae_s^1, t_s^1 \implies \mathsf{pdf}[\![ae_1]\!] = \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![t_s^1]\!]$$

Because $\mathcal{S} \vdash ae_1 \Rrightarrow_{ex} ae_s^1, t_s^1$,

$$\mathsf{pdf}[\![ae_1]\!] = \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![t_s^1]\!]$$

By induction hypothesis

$$\mathcal{S} \vdash ae_2 \Rrightarrow_{ex} ae_s^2, t_s^2 \implies \mathsf{pdf}[\![ae_2]\!] = \mathsf{pdf}[\![ae_s^2]\!] * \mathsf{pdf}[\![t_s^2]\!]$$

Because $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^2, t_s^2$

$$\mathsf{pdf}[\![ae_2]\!] = \mathsf{pdf}[\![ae_s^2]\!] * \mathsf{pdf}[\![t_s^2]\!]$$

From definition of pdf

$$\mathsf{pdf}[\![ae]\!] = \mathsf{pdf}[\![ae_1]\!] * \mathsf{pdf}[\![ae_2]\!]$$

$$= \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![ae_s^2]\!] * \mathsf{pdf}[\![t_s^1]\!] * \mathsf{pdf}[\![t_s^2]\!]$$

From definition of pdf $\mathsf{pdf}[\![ae_s]\!] = \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![ae_s^2]\!]$ and $\mathsf{pdf}[\![t_s]\!] = \mathsf{pdf}[\![t_s^1]\!] * \mathsf{pdf}[\![t_s^2]\!]$. Therefore

$$\mathsf{pdf}[\![ae]\!] = \mathsf{pdf}[\![ae_s]\!] * \mathsf{pdf}[\![t_s]\!]$$

Case 2: $ae = ((ae_1\ ae_2)x = ae_3 : v)\#id$ and $ID(ae_1) \in \mathcal{S}$
By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash (ae_1\ ae_2)x = ae_3 : v \Rightarrow_{ex} (ae_s^1\ ae_s^2)x = ae_3 : v, t_s^1; t_s^2$$

Then $ae_s = (ae_s^1\ ae_s^2)x = ae_3 : v$, $t_s = t_s^1; t_s^2$, $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$, and $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^2, t_s^2$.
By induction hypothesis

$$\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1 \implies \mathsf{pdf}[\![ae_1]\!] = \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![t_s^1]\!]$$

Because $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$,

$$\mathsf{pdf}[\![ae_1]\!] = \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![t_s^1]\!]$$

By induction hypothesis

$$\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^2, t_s^2 \implies \mathsf{pdf}[\![ae_2]\!] = \mathsf{pdf}[\![ae_s^2]\!] * \mathsf{pdf}[\![t_s^2]\!]$$

Because $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^2, t_s^2$

$$\mathsf{pdf}[\![ae_2]\!] = \mathsf{pdf}[\![ae_s^2]\!] * \mathsf{pdf}[\![t_s^2]\!]$$

From definition of pdf

$$\mathsf{pdf}[\![ae]\!] = \mathsf{pdf}[\![ae_1]\!] * \mathsf{pdf}[\![ae_2]\!] * \mathsf{pdf}[\![ae_3]\!]$$

$$= \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![ae_s^2]\!] * \mathsf{pdf}[\![ae_3]\!] * \mathsf{pdf}[\![t_s^1]\!] * \mathsf{pdf}[\![t_s^2]\!]$$

From definition of pdf $\mathsf{pdf}[\![ae_s]\!] = \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![ae_s^2]\!] * \mathsf{pdf}[\![ae_3]\!]$ and $\mathsf{pdf}[\![t_s]\!] = \mathsf{pdf}[\![t_s^1]\!] * \mathsf{pdf}[\![t_s^2]\!]$. Therefore

$$\mathsf{pdf}[\![ae]\!] = \mathsf{pdf}[\![ae_s]\!] * \mathsf{pdf}[\![t_s]\!]$$

Case 3: $ae = ((ae_1\ ae_2)x = ae_3 : v)\#id$ and $ID(ae_1) \notin \mathcal{S}$

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash (ae_1\ ae_2)x = ae_3 : v \Rightarrow_{ex} ae_s^3 : v, t_s^1; \mathsf{assume}\ y = ae_s^1; t_s^2; \mathsf{assume}\ x = ae_s^2; t_s^3$$

Then $ae_s = ae_s^3 : v$, $t_s = t_s^1; \mathsf{assume}\ y = ae_s^1; t_s^2; \mathsf{assume}\ x = ae_s^2; t_s^3$, $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$, $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^2, t_s^2$, $\mathcal{S} \vdash ae_3 \Rightarrow_{ex} ae_s^3, t_s^3$.

By induction hypothesis

$$\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1 \implies \mathsf{pdf}[\![ae_1]\!] = \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![t_s^1]\!]$$

Because $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$,

$$\mathsf{pdf}[\![ae_1]\!] = \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![t_s^1]\!]$$

By induction hypothesis

$$\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^2, t_s^2 \implies \mathsf{pdf}[\![ae_2]\!] = \mathsf{pdf}[\![ae_s^2]\!] * \mathsf{pdf}[\![t_s^2]\!]$$

Because $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^2, t_s^2$

$$\mathsf{pdf}[\![ae_2]\!] = \mathsf{pdf}[\![ae_s^2]\!] * \mathsf{pdf}[\![t_s^2]\!]$$

By induction hypothesis

$$\mathcal{S} \vdash ae_3 \Rightarrow_{ex} ae_s^3, t_s^3 \implies \mathsf{pdf}[\![ae_3]\!] = \mathsf{pdf}[\![ae_s^3]\!] * \mathsf{pdf}[\![t_s^3]\!]$$

Because $\mathcal{S} \vdash ae_3 \Rightarrow_{ex} ae_s^3, t_s^3$

$$\mathsf{pdf}[\![ae_3]\!] = \mathsf{pdf}[\![ae_s^3]\!] * \mathsf{pdf}[\![t_s^3]\!]$$

From definition of pdf

$$\mathsf{pdf}[\![ae]\!] = \mathsf{pdf}[\![ae_1]\!] * \mathsf{pdf}[\![ae_2]\!] * \mathsf{pdf}[\![ae_3]\!]$$

$$= \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![ae_s^2]\!] * \mathsf{pdf}[\![ae_s^3]\!] * \mathsf{pdf}[\![t_s^1]\!] * \mathsf{pdf}[\![t_s^2]\!] * \mathsf{pdf}[\![t_s^3]\!]$$

From definition of pdf $\mathsf{pdf}[\![ae_s]\!] = \mathsf{pdf}[\![ae_s^3]\!]$ and $\mathsf{pdf}[\![t_s]\!] = \mathsf{pdf}[\![t_s^1]\!] * \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![t_s^2]\!] * \mathsf{pdf}[\![ae_s^2]\!] * \mathsf{pdf}[\![t_s^3]\!]$. Therefore

$$\mathsf{pdf}[\![ae]\!] = \mathsf{pdf}[\![ae_s]\!] * \mathsf{pdf}[\![t_s]\!]$$

Case 4: $ae = \mathsf{Dist}(ae_1 \# id_e) = ae_2 : v$ and $id_e \in \mathcal{S}$

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash \mathsf{Dist}(ae_1) = ae_2 : v \Rightarrow_{ex} \mathsf{Dist}(ae_s^1) = ae_2 : v, t_s^1$$

Then $ae_s = \mathsf{Dist}(ae_1) = ae_2 : v$, $t_s = t_s^1$, and $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$.

By induction hypothesis

$$\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1 \implies \mathsf{pdf}[\![ae_1]\!] = \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![t_s^1]\!]$$

Because $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$,

$$\mathsf{pdf}[\![ae_1]\!] = \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![t_s^1]\!]$$

From definition of pdf, $ae_2 \Rightarrow_r e$

$$\mathsf{pdf}[\![ae]\!] = \mathsf{pdf}[\![ae_1]\!] * \mathsf{pdf}[\![ae_2]\!] * \mathsf{pdf}_{\mathsf{Dist}}(\mathcal{V}(ae_1), e)$$

$$= \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![ae_2]\!] * \mathsf{pdf}[\![t_s^1]\!] * \mathsf{pdf}_{\mathsf{Dist}}(\mathcal{V}(ae_1), e)$$

From definition of pdf $\mathsf{pdf}[\![ae_s]\!] = \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![ae_2]\!] * \mathsf{pdf}_{\mathsf{Dist}}(\mathcal{V}(ae_1), e)$ and $\mathsf{pdf}[\![t_s]\!] = \mathsf{pdf}[\![t_s^1]\!]$. Therefore

$$\mathsf{pdf}[\![ae]\!] = \mathsf{pdf}[\![ae_s]\!] * \mathsf{pdf}[\![t_s]\!]$$

Case 5: $ae = (\mathsf{Dist}(ae_1\#id_e) = ae_2 : v)\#id$ and $id_e \notin \mathcal{S}$

By assumption

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash \mathsf{Dist}(ae_1) = ae_2 : v \Rightarrow_{ex} ae_s^2 : v, t_s^1; \mathsf{observe}(\mathsf{Dist}(ae_s^1) = e_v); t_s^2$$

Then $ae_s = ae_s^2 : v$, $t_s = t_s^1$, $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$, $ae_2 \Rightarrow_r e_v$and $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^1, t_s^1$.

By induction hypothesis

$$\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1 \implies \mathsf{pdf}[\![ae_1]\!] = \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![t_s^1]\!]$$

Because $\mathcal{S} \vdash ae_1 \Rightarrow_{ex} ae_s^1, t_s^1$,

$$\mathsf{pdf}[\![ae_1]\!] = \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![t_s^1]\!]$$

By induction hypothesis

$$\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^2, t_s^2 \implies \mathsf{pdf}[\![ae_2]\!] = \mathsf{pdf}[\![ae_s^2]\!] * \mathsf{pdf}[\![t_s^2]\!]$$

Because $\mathcal{S} \vdash ae_2 \Rightarrow_{ex} ae_s^2, t_s^2$,

$$\mathsf{pdf}[\![ae_2]\!] = \mathsf{pdf}[\![ae_s^2]\!] * \mathsf{pdf}[\![t_s^2]\!]$$

From definition of pdf,

$$\mathsf{pdf}[\![ae]\!] = \mathsf{pdf}[\![ae_1]\!] * \mathsf{pdf}[\![ae_2]\!] * \mathsf{pdf}_{\mathsf{Dist}}(\mathcal{V}(ae_1), e)$$

$$= \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![ae_s^2]\!] * \mathsf{pdf}[\![t_s^1]\!] * \mathsf{pdf}[\![t_s^2]\!] * \mathsf{pdf}_{\mathsf{Dist}}(\mathcal{V}(ae_1), e)$$

From definition of pdf $\mathsf{pdf}[\![ae_s]\!] = \mathsf{pdf}[\![ae_s^2]\!]$, $\mathsf{pdf}[\![t_s]\!] = \mathsf{pdf}[\![t_s^1]\!] * \mathsf{pdf}[\![ae_s^1]\!] * \mathsf{pdf}[\![t_s^2]\!] * \mathsf{pdf}_{\mathsf{Dist}}(\mathcal{V}(ae_s^1), e)$ and $\mathcal{V}(ae_s^1) = \mathcal{V}(ae_1)$ (Observation 2). Therefore

$$\mathsf{pdf}[\![ae]\!] = \mathsf{pdf}[\![ae_s]\!] * \mathsf{pdf}[\![t_s]\!]$$

Because we have considered all cases, by induction, for augmented expression ae, subproblem $\mathcal{S}$, augmented subexpression ae_s, and a subtrace t_s,

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s \implies \mathsf{pdf}[\![ae]\!] = \mathsf{pdf}[\![ae_s]\!] * \mathsf{pdf}[\![t_s]\!]$$

□

Theorem 9. *Given a trace t and a valid subproblem $\mathcal{S}$ on t, then for subtrace $t_s =$ $\mathsf{ExtractTrace}(t, \mathcal{S})$,*

$$\mathsf{pdf}[\![t]\!] = \mathsf{pdf}[\![t_s]\!]$$

i.e. for the unnormalized density of t and t_s is equal.

Proof. Proof by induction

Base Case: $t = \emptyset$

By assumption

$$\mathcal{S} \vdash t \Rightarrow_{ex} t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash \emptyset \Rightarrow_{ex} \emptyset$$

Then $t_s = \emptyset$.

By definition of pdf, $\mathsf{pdf}[\![\emptyset]\!] = 1$

$$\mathsf{pdf}[\![t]\!] = \mathsf{pdf}[\![t_s]\!]$$

Induction Case:

Case 1: $t = \mathsf{assume}\ x = ae; t_1$

By assumption

$$\mathcal{S} \vdash t \Rightarrow_{ex} t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash \mathsf{assume}\ x = ae; t_1 \Rightarrow_{ex} t_s^1; \mathsf{assume}\ x = ae_s; t_s^2$$

Then $\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s^1$, $\mathcal{S} \vdash t_1 \Rightarrow_{ex} t_s^2$.

By induction hypothesis

$$\mathcal{S} \vdash t_1 \Rightarrow_{ex} t_s^2 \implies \mathsf{pdf}[\![t_1]\!] = \mathsf{pdf}[\![t_s^2]\!]$$

Because $\mathcal{S} \vdash t_1 \Rightarrow_{ex} t_s^2$,

$$\mathsf{pdf}[\![t_1]\!] = \mathsf{pdf}[\![t_s^2]\!]$$

From Lemma 14 over augmented expressions

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s^1 \implies \mathsf{pdf}[\![ae]\!] = \mathsf{pdf}[\![ae_s]\!] * \mathsf{pdf}[\![t_s^1]\!]$$

Because $\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s^1$

$$\mathsf{pdf}[\![ae]\!] = \mathsf{pdf}[\![ae_s]\!] * \mathsf{pdf}[\![t_s^1]\!]$$

From definition of pdf

$$\mathsf{pdf}[\![t]\!] = \mathsf{pdf}[\![ae]\!] * \mathsf{pdf}[\![t_1]\!] = \mathsf{pdf}[\![ae_s]\!] * \mathsf{pdf}[\![t_s^1]\!] * \mathsf{pdf}[\![t_s^2]\!]$$

Because $\mathsf{pdf}[\![t_s]\!] = \mathsf{pdf}[\![ae_s]\!] * \mathsf{pdf}[\![t_s^1]\!] * \mathsf{pdf}[\![t_s^2]\!]$

$$\mathsf{pdf}[\![t]\!] = \mathsf{pdf}[\![t_s]\!]$$

Case 2: $t = \mathsf{observe}(\mathsf{Dist}(ae) = e); t_1$

By assumption

$$\mathcal{S} \vdash t \Rightarrow_{ex} t_s$$

By definition of $\Rightarrow_{ex}$

$$\mathcal{S} \vdash \mathsf{observe}(\mathsf{Dist}(ae) = e); t_1 \Rightarrow_{ex} t_s^1; \mathsf{observe}(\mathsf{Dist}(ae_s) = e); t_s^2$$

Then $\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s^1$, $\mathcal{S} \vdash t_1 \Rightarrow_{ex} t_s^2$.

By induction hypothesis

$$\mathcal{S} \vdash t_1 \Rightarrow_{ex} t_s^2 \implies \mathsf{pdf}[\![t_1]\!] = \mathsf{pdf}[\![t_s^2]\!]$$

Because $\mathcal{S} \vdash t_1 \Rightarrow_{ex} t_s^2$,

$$\mathsf{pdf}[\![t_1]\!] = \mathsf{pdf}[\![t_s^2]\!]$$

From Lemma 14 over augmented expressions

$$\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s^1 \implies \mathsf{pdf}[\![ae]\!] = \mathsf{pdf}[\![ae_s]\!] * \mathsf{pdf}[\![t_s^1]\!]$$

Because $\mathcal{S} \vdash ae \Rightarrow_{ex} ae_s, t_s^1$

$$\mathsf{pdf}[\![ae]\!] = \mathsf{pdf}[\![ae_s]\!] * \mathsf{pdf}[\![t_s^1]\!]$$

From definition of pdf

$$\mathsf{pdf}[\![t]\!] = \mathsf{pdf}[\![ae]\!]*\mathsf{pdf}[\![t_1]\!]*\mathsf{pdf}_{\mathsf{Dist}}(\mathcal{V}(ae), e) = \mathsf{pdf}[\![ae_s]\!]*\mathsf{pdf}[\![t_s^1]\!]*\mathsf{pdf}[\![t_s^2]\!]*\mathsf{pdf}_{\mathsf{Dist}}(\mathcal{V}(ae), e)$$

Because $\mathsf{pdf}[\![t_s]\!] = \mathsf{pdf}[\![ae_s]\!] * \mathsf{pdf}[\![t_s^1]\!] * \mathsf{pdf}[\![t_s^2]\!] * \mathsf{pdf}_{\mathsf{Dist}}(\mathcal{V}(ae), e)$ and $\mathcal{V}(ae) = \mathcal{V}(ae_s)$ (Observation 2)

$$\mathsf{pdf}[\![t]\!] = \mathsf{pdf}[\![t_s]\!]$$

Because we have covered all cases, by induction, for any trace t and subproblem $\mathcal{S}$

$$\mathcal{S} \vdash t \Rightarrow_{ex} t_s \implies \mathsf{pdf}[\![t]\!] = \mathsf{pdf}[\![t_s]\!]$$

Therefore for any trace t and subproblem $\mathcal{S}$

$$t_s = \mathsf{ExtractTrace}(t, \mathcal{S}) \implies \mathsf{pdf}[\![t]\!] = \mathsf{pdf}[\![t_s]\!]$$

□

Consider an equivalence class c_i and partitioned trace space T_{c_i}, let p_s be the subprogram, such that all subtraces of traces in T_{c_i} are valid traces of p_s. (T_{p_s}, Σ_{p_s}) is the measurable space over traces of subprogram p_s. The normalized probability distribution $\mu_{p_s} : \Sigma_{p_s} \to [0,1]$ for the subprogram p_s is

$$\mu_{p_s}(A) = \frac{\sum\limits_{t_s \in A} \mathsf{pdf}[\![t_s]\!]}{\sum\limits_{t_s \in T_{p_s}} \mathsf{pdf}[\![t_s]\!]} = \frac{\sum\limits_{t \in A'} \mathsf{pdf}[\![t]\!]}{\sum\limits_{t \in T_{c_i}} \mathsf{pdf}[\![t]\!]} = (\mu_p)_i(A') = \frac{\mu_p(A')}{\mu_p(T_{c_i})} = v_i(c_i, A')$$

where $A' = \left\{t' \middle| t_s \in A, t' = \mathsf{StitchTrace}(t, t_s, \mathsf{SS}(t))\right\}$ for any trace $t \in T_{c_i}$.

Hence, sampling/inference over subprogram p_s is equivalent to sampling/inference over the original program p with the constraint that all traces belong to the equivalence class c_i.

Theorem 10. *Consider equivalence class c_i and partitioned trace space T_{c_i}, let p_s be the subprogram, such that all subtraces of traces in T_{c_i} are valid traces of p_s. Then given a Markov Kernel $K : T_{p_s} \times \Sigma_{p_s} \to [0,1]$ which is μ_{p_s}-irreducible, aperiodic and with stationary distribution μ_{p_s}, the Markov kernel $K(c_i) : T_{c_i} \times \Sigma_{c_i} \to [0,1]$ defined as*

$$K(c_i)(t, A) = K(t_s, A')$$

where $t_s = \mathsf{ExtractTrace}(t, \mathsf{SS}(t))$ and $A' = \{t_s | t \in A, t_s = \mathsf{ExtractTrace}(t, \mathsf{SS}(t))\}$, is $v_i(c_i, .)$-irreducible, aperiodic and with stationary distribution $v_i(c_i, .)$.

Proof. ExtractTrace is one-one function from T_{c_i} to T_{p_s} and push forward measure of $v_i(c_i, .)$ is μ_{p_s}. □

5.1.5 Generalized Markov Kernels

Definition 23 (Generalized Markov Kernel). *A Generalized Markov Kernel K is a parameterized Markov Kernel which when parameterized with a probabilistic program p, which defines the probability space (T_p, Σ_p, μ_p) (as defined above), returns a Markov Kernel $K(p) : T_p \times \Sigma_p \to [0,1]$, which is μ_p-irreducible, aperiodic and has the stationary distribution μ_p.*

A generalized Markov Kernel formalizes the concept of Markov chain inference algorithms. The inference algorithms used within the probabilistic programming framework are generally coded to work with any input probabilistic program p and still provide convergence guarantees. For example, Venture [24] allows the programmer to use a variety of inference algorithms which in general work on all probabilistic programs which can be written in that language.

Definition 24 (Generalized Class Kernels). *Given a generalized Markov Kernel K and a subproblem selection strategy SS, a generalized class Kernel $K_{f_{\mathsf{SS}}}$ is a parameterized with a probabilistic program p (which defines space (T_p, Σ_p, μ_p)), where*

$$K_{f_{\mathsf{SS}}}(p)(t, A) = K'(c_i)(t, A) = K(p_s)(t_s, A')$$

where $t_s = \mathsf{ExtractTrace}(t, \mathsf{SS}(t))$, $p_s = \mathsf{Program}(t_s)$, $f_{\mathsf{SS}}(t) = \langle c_i, t\rangle$ and $A' = \{t_s | t \in A, t_s = \mathsf{ExtractTrace}(t, \mathsf{SS}(t))\}$.

5.2 Inference Metaprogramming

Using the concept of independent subproblem inference (Chapter 3) and generalized Markov Kernels, I define an Inference Metaprogramming Language (Figure 5-2). An inference metaprogram is either one of the black box Generalized Markov Kernel inference algorithms $b_i : T_p \to T_p$ in my framework, which takes a trace from an arbitrary program p as an input and returns another trace from the same program, or a finite set $S = \{p_1\ ic_1, p_2\ ic_2 \ldots p_k\ ic_k\}$ of inference statements with an attached probability value $p_i \in (0, 1)$, such that $\sum\limits_{i=0}^{k} p_k = 1$. These probability values are used to randomly select a subproblem inference statement to execute. Each infer statement is parameterized with a subproblem selection strategy SS, which returns a valid subproblem over input trace t and an inference metaprogram that is executed over the subtrace. Figure 5-3 presents the execution semantics of my inference metaprogramming language. In comparison with entangled subproblem inference, one benefit of the approach is that it is straightforward to apply independent subproblem inference recursively.

Theorem 11. *If all the subproblems used in my inference metaprograms are reversible and connect the space of their respective input probabilistic programs, then all inference*

$$\begin{array}{rcl} ic \in IC & := & \mathsf{infer}(SS, ip) \\ ip \in IP & := & b_i | \{p_1\ ic_1, p_2\ ic_2 \dots p_k\ ic_k\} \quad \mathsf{where} \quad \sum_{i=0}^{k} p_k = 1 \end{array}$$

Figure 5-2: Inference Metaprogramming language

$$\frac{t' = b(t)}{b, t \Rightarrow_i t'} \qquad \frac{\begin{array}{c} n \sim \mathsf{multinomial}(p_1, p_2 \dots p_k) \quad ic_n = \mathsf{infer}(SS_n, ip_n) \quad SS_n(t) = \mathcal{S} \\ t_s = \mathsf{ExtractTrace}(t, \mathcal{S}) \quad ip_n, t_s \Rightarrow_i t'_s \quad t' = \mathsf{StitchTrace}(t, t'_s, \mathcal{S}) \\ \mathsf{where}\ \ t'_s \in \mathsf{Traces}(\mathsf{Program}(t_s)) \end{array}}{\{p_1\ ic_1, p_2\ ic_2 \dots p_k\ ic_k\}, t \Rightarrow_i t'}$$

Figure 5-3: Execution Semantics for Inference Metaprograms

metaprograms in my inference metaprogramming language implement a generalized Markov Kernel.

Proof. Proof by induction over structure of inference metaprograms,

Base Case: All black box inference algorithms in my inference metaprogramming language are generalized Markov Kernels. Hence given traces of program p (which define probability space (T_p, Σ_p, μ_p)) the black box inference algorithm is μ_p-irreducible, aperiodic and has the stationary distribution μ_p.

Induction Case: Consider the inference metaprogram $ip = \{p_1\ ic_1, p_2, ic_2, \dots p_k\ ic_k\}$, where $ic_i = \mathsf{infer}(\mathsf{SS}_i, ip_i)$ and $\sum_{i=1}^{k} p_i = 1$.

Using induction hypothesis, I assume, for all $i \in \{1, 2, \dots k\}$, all ip_i implement a Generalized Markov Kernel K^{ip_i}. Since my subproblem SS is reversible, I lift the Generalized Markov Kernel to Generalized Class Kernel (Definition 24) $K_{f_{SS_i}}$, where for any program p (which defines space (T_p, Σ_p, μ_p)), defines a class kernel.

Given an probabilistic program p, the inference metaprogram ip implements the Generalized Markov Kernel K, where

$$K(p)(t, A) = \sum_{i=1}^{k} p_i K_{f_{SS_i}}(t, A)$$

Using Theorems 4, 5, and 6, if the class functions $f_{SS_1}, f_{SS_2}, \dots f_{SS_k}$ connect the space (T_p, Σ_p, μ_p), $K(p)$ is μ_p-irreducible, aperiodic and has μ_p as its stationary distribution.

□

Corollary 2. *Given an input probabilistic program p (defining trace space (T_p, Σ_p, μ_p)), inference metaprograms which use reversible subproblem selection strategies which connect the space of their respective input probabilistic programs, will converge to μ_p.*

Chapter 6

Related Work

I discuss related work in three areas: probabilisitic programming languages, subproblem inference in probabilistic programming, and asymptotic convergence.

Probabilistic Programming Languages: Over the last several decades researchers have developed a range of probabilistic programming languages. With current practice each language typically comes paired with one/a few black box inference strategies. Example language/inference pairs include Stan [5] with Hamiltonian Monte Carlo inference [1]; Anglician [38] with particle Gibbs, etc. Languages like LibBi [29], Edward [39] and Pyro [20] provide inference customization mechanisms, but without subproblems or asymptotic convergence guarantees.

The Augur [18, 40] compiler generates efficient compiled implementations of Markov Chain Monte Carlo inference algorithms that operate over a fixed set of stochastic choices. Our techniques, in contrast, work with traces that have a dynamically changing and potentially unbounded set of stochastic choices.

Subproblem Inference: Both Turing [10] and Venture [24] provide inference metaprogramming constructs with subproblems and different inference algorithms that operate on these subproblems. In Venture subproblem inference is performed over full program traces, with subproblems entangled with the full trace. The inference algorithms in Venture must therefore operate over the entire trace while ensuring that the inference effects do not escape the specified subproblem. Our extraction and stitching technique eliminates this entanglement and enables the use of standard inference algorithms that operate over complete traces while still supporting subproblem identification and inference. Turing only provides mechanisms that target specific stochastic choices in the context of the complete probabilistic computation.

Asymptotic Convergence: There is a vast literature on asymptotic convergence of Markov chain algorithms in various statistics and probability settings [25, 37]. Our work is unique in that it provides the first characterization of asymptotic convergence for subproblem inference in probabilistic programs. Complications that occur in this setting include mixtures of discrete and continuous variables, stochastic choices with cascading effects that may change the number of stochastic choices in the computation, and resulting sample spaces with unbounded numbers of random variables. Standard results from computational statistics, computational physics, and Monte-Carlo methods focus on finite dimensional discrete state spaces, a context in which linear algebra (i.e., spectral analysis of the transition matrix of the underlying Markov chain [8] or coupling arguments [21]) is sufficient to prove convergence. State spaces with continuous random variables are outside the scope of these formal analyses. Measure-theoretic treatments are more general [31]. Our results show how to apply the concepts in these treatments to prove asymptotic convergence results in my probabilistic programming with subproblems contexts.

Bibliography

[1] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1-2):5–43, 2003.

[2] Krishna B Athreya, Hani Doss, Jayaram Sethuraman, et al. On the convergence of the markov chain simulation method. *The Annals of Statistics*, 24(1):69–100, 1996.

[3] Eric Atkinson and Michael Carbin. Typesafety for explicitly-coded probabilistic inference procedures. 2017.

[4] Johannes Borgström, Ugo Dal Lago, Andrew D Gordon, and Marcin Szymczak. A lambda-calculus foundation for universal probabilistic programming. In *ACM SIGPLAN Notices*, volume 51, pages 33–46. ACM, 2016.

[5] Bob Carpenter, Andrew Gelman, Matt Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Michael A. Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software*, 20:1–37, 2016.

[6] Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335, 1995.

[7] Ugo Dal Lago and Margherita Zorzi. Probabilistic operational semantics for the lambda calculus. *RAIRO-Theoretical Informatics and Applications*, 46(3):413–450, 2012.

[8] Persi Diaconis and Daniel Stroock. Geometric bounds for eigenvalues of markov chains. *The Annals of Applied Probability*, pages 36–61, 1991.

[9] David A Forsyth and Jean Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[10] Hong Ge, Kai Xu, and Zoubin Ghahramani. Turing: Composable inference for probabilistic programming. In *International Conference on Artificial Intelligence and Statistics*, pages 1682–1690, 2018.

[11] Timon Gehr, Sasa Misailovic, and Martin Vechev. Psi: Exact symbolic inference for probabilistic programs. In *International Conference on Computer Aided Verification*, pages 62–83. Springer, 2016.

[12] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*, volume 2. CRC press Boca Raton, FL, 2014.

[13] Charles J Geyer. Markov chain monte carlo lecture notes. *Course notes, Spring Quarter*, 1998.

[14] Noah Goodman, Vikash Mansinghka, Daniel M Roy, Keith Bonawitz, and Joshua B Tenenbaum. Church: a language for generative models. In *Proceedings of the 25th International Conference on Uncertainty in Artificial Intelligence*, 2008.

[15] Noah D. Goodman and Andreas Stuhlmueller. The Design and Implementation of Probabilistic Programming Languages. `http://dippl.org`, 2014. Accessed: 2017-11-15.

[16] Andrew D. Gordon, Thore Graepel, Nicolas Rolland, Claudio Russo, Johannes Borgstrom, and John Guiver. Tabular: a schema-driven probabilistic programming language. In *ACM SIGPLAN Notices*, volume 49, pages 321–334. ACM, 2014.

[17] Andrew D. Gordon, Thomas A. Henzinger, Aditya V. Nori, and Sriram K. Rajamani. Probabilistic programming. In *Proceedings of the on Future of Software Engineering*, pages 167–181. ACM, 2014.

[18] Daniel Huang, Jean-Baptiste Tristan, and Greg Morrisett. Compiling markov chain monte carlo algorithms for probabilistic modeling. In *ACM SIGPLAN Notices*, volume 52, pages 111–125. ACM, 2017.

[19] Dexter Kozen. Semantics of probabilistic programs. In *Foundations of Computer Science, 1979., 20th Annual Symposium on*, pages 101–114. IEEE, 1979.

[20] Uber AI Labs. Pyro, a deep probabilistic programming language, 2017.

[21] David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

[22] Jun S Liu. *Monte Carlo strategies in scientific computing.* Springer Science & Business Media, 2008.

[23] Vikash Mansinghka, Daniel Selsam, and Yura Perov. Venture: a higher-order probabilistic programming platform with programmable inference. *arXiv preprint arXiv:1404.0099*, 2014.

[24] Vikash K Mansinghka, Ulrich Schaechtle, Shivam Handa, Alexey Radul, Yutian Chen, and Martin Rinard. Probabilistic programming with programmable inference. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 603–616. ACM, 2018.

[25] Sean P Meyn and Richard L Tweedie. *Markov chains and stochastic stability.* Springer Science & Business Media, 2012.

[26] Brian Milch, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. Blog: Probabilistic models with unknown objects. *Statistical relational learning*, page 373, 2007.

[27] David Monniaux. An abstract monte-carlo method for the analysis of probabilistic programs. *arXiv preprint cs/0701195*, 2007.

[28] Kevin P Murphy. *Machine learning: a probabilistic perspective.* MIT press, 2012.

[29] Lawrence M Murray. Bayesian state-space modelling on high-performance hardware using libbi. *arXiv preprint arXiv:1306.3277*, 2013.

[30] Norman Ramsey and Avi Pfeffer. Stochastic lambda calculus and monads of probability distributions. In *ACM SIGPLAN Notices*, volume 37, pages 154–165. ACM, 2002.

[31] Gareth O Roberts, Jeffrey S Rosenthal, et al. General state space markov chains and mcmc algorithms. *Probability surveys*, 1:20–71, 2004.

[32] Gareth O Roberts and Adrian FM Smith. Simple conditions for the convergence of the gibbs sampler and metropolis-hastings algorithms. *Stochastic processes and their applications*, 49(2):207–216, 1994.

[33] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education, 2 edition, 2003.

[34] Micha Sharir, Amir Pnueli, and Sergiu Hart. Verification of probabilistic programs. *SIAM Journal on Computing*, 13(2):292–314, 1984.

[35] Sam Staton, Hongseok Yang, Frank Wood, Chris Heunen, and Ohad Kammar. Semantics for probabilistic programming: higher-order functions, continuous distributions, and soft constraints. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 525–534. ACM, 2016.

[36] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics.* MIT press, 2005.

[37] Luke Tierney. Markov chains for exploring posterior distributions. *the Annals of Statistics*, pages 1701–1728, 1994.

[38] David Tolpin, Jan-Willem van de Meent, and Frank Wood. Probabilistic programming in anglican. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 308–311. Springer, 2015.

[39] Dustin Tran, Matthew D Hoffman, Rif A Saurous, Eugene Brevdo, Kevin Murphy, and David M Blei. Deep probabilistic programming. *arXiv preprint arXiv:1701.03757*, 2017.

[40] Jean-Baptiste Tristan, Daniel Huang, Joseph Tassarotti, Adam C. Pocock, Stephen Green, and Guy L. Steele. Augur: Data-parallel probabilistic modeling. In *Advances in Neural Information Processing Systems*, pages 2600–2608, 2014.