Theory Notes for Machine Theory of Mind

Date: 2025-12-03

Repo: Machine-Theory-Of-Mind

LaTeX appendix: `docs/theory_appendix.tex` (linked as Appendix A in the main paper)

1. Bayesian Consistency for Discrete Mental-State Hypotheses

Statement 1.1 (Posterior Concentration). Let H = {h_1, \ldots, h_m} be a finite hypothesis space over mental-state descriptors (warmth/competence mixtures, observer segments, etc.). Suppose data (O_t)_{t \ge 1} is generated i.i.d. from the true hypothesis h*. If (i) the model is identifiable in the sense that for every h \neq h* there exists some observation o with P_{h}(o) \neq P_{h*}(o), and (ii) P_{h}(o) > 0 for all h in H and all feasible o observed under h*, then for any prior with full support on H the posterior mass satisfies

```
lim_{t -> infinity} P(h* | O_1:t) = 1 almost surely.
```

Lemma 1.2 (Likelihood Separation). Under Assumption (i), there exists c(h) > 0 such that for every competing hypothesis h \neq h*

```
limsup_{t -> infinity} (1 / t) log [ P(O_1:t | h) / P(O_1:t | h*) ] <= -c(h).
```

Proof sketch. The lemma follows from the strong law of large numbers applied to log-likelihood ratios with finite alphabet observations. Nonzero likelihoods rule out divisions by zero, while identifiability makes the expected log-ratio negative for h \neq h*. Summing the finite number of competitors bounds the posterior denominator, yielding almost-sure concentration on h* by standard Bayesian consistency (e.g., Doob, 1949). For our agents, H can index discrete warmth/competence bins inside `src/models/bayesian_mental_state.py`, so the conditions reduce to requiring that every bin retains strictly positive weight in the prior and the observation channel used in `BayesianMentalState.bayesian_update` never assigns zero likelihood to realised bins.

Experimental hook. `experiments/config/week5_bayesian_sweep.yaml` and `src/experiments/run_week4.py` already log posterior trajectories; to verify the statement we can log likelihood ratios per hypothesis ID and confirm exponential decay for rejected bins during the week3/4 replay ablations archived in `docs/internal/results/week3/analysis_report.md` and `docs/internal/results/week4/analysis_report.md`.

2. Social Intelligence as Multi-Objective Optimization

Let pi denote any agent policy deployed through `src/agents`. Define

```
R(pi) = E_pi[task_reward],    S(pi) = E_pi[SIQ(records)],
```

where `SIQ` is computed with `src/metrics/siq.py` over the episodic records produced by the trace runner. The design goal is to maximize the vector objective J(pi) = (R(pi), S(pi)).

Definition 2.1 (Pareto Dominance). Policy pi_a dominates pi_b when R(pi_a) \ge R(pi_b) and S(pi_a) \ge S(pi_b) with at least one strict inequality. A policy is Pareto optimal if no other policy dominates it.

Proposition 2.2 (Scalarization Sufficiency). Suppose SIQ weights stay positive as in `SIQConfig`. Then every Pareto-optimal policy is a solution of the scalarized objective

```
max_pi  (1 - lambda) R(pi) + lambda S(pi)
```

for some lambda in [0, 1], and conversely every optimizer of the scalarized problem is Pareto optimal.

Proof sketch. Standard convex multi-objective arguments apply once SIQ is treated as a smooth expectation over episodic traces. Positivity of SIQ weights ensures strict monotonicity, preventing flat faces where scalarization could miss extreme points. In practice lambda corresponds to `lambda_social` in `src/agents/bayesian_mtom_agent.py`, with S(pi) approximated by the logged SIQ components stored by `demo/trace_dashboard.py`. Choosing lambda from `{0.1, 0.3, 0.5, 0.7}` traces the empirical Pareto frontier when we run `experiments/run_experiment.py` or the week7 trace sweeps.

Experimental hook. `experiments/run_trace_sweep_extended.py` already sweeps opponent policies; adding a lambda grid and logging `(R, S)` pairs lets us build Pareto plots in `results/week7/plots`. Scripts `results/week5/final_combined_report.md` and `results/week5/stats_summary.json` provide templates for summarizing dominated policies.

3. First-Order SocialScore Improvement Under Small lambda

Let `SocialScore` be the evaluator in `src/social/social_score.py`. Consider a policy family pi_lambda that maximizes the entropy-regularized objective used in `BayesianSocialScorer.bayesian_utility`:

```
J_lambda(pi) = E_pi[ task_reward(a) ] + lambda E_pi[ Delta_obs(a) ] - tau KL(pi || pi_re
```

where Delta_obs(a) is the expected observer response predicted by `BayesianSocialScorer` and tau > 0 is the implicit temperature created by the risk penalty in `BayesianMToMAgent.make_offer`.

Lemma 3.1 (Correct Observer Model). If the predictive distribution returned by `BayesianSocialScorer.predict_perception_distribution` matches the true observer, then Delta_obs(a) equals the marginal change in logged `social_score` for action a up to a fixed normalization constant kappa > 0 determined by the `LinearSocialScore` weights.

Theorem 3.2 (First-Order Gain for Small lambda). Assume: (i) the optimizer pi_lambda is differentiable at lambda = 0 under the entropy-regularized objective above, (ii) there exists at least one action a with Delta_obs(a) > Delta_obs mean under pi_0, and (iii) Lemma 3.1 holds. Then for sufficiently small lambda > 0,

```
SocialScore(pi_lambda) >= SocialScore(pi_0) + (lambda / tau) Var_{a ~ pi_0}[Delta_obs(a
```

In particular SocialScore(pi_lambda) > SocialScore(pi_0) whenever the variance term is nonzero and lambda lies below the radius where the quadratic remainder dominates.

Proof sketch. Under entropy-regularized best responses, pi_lambda has the closed form `pi_lambda(a) proportional to pi_0(a) exp(lambda Delta_obs(a) / tau)` (mirror descent / logit response). Taking the derivative at lambda = 0 yields `d/dlambda E_{pi_lambda}[Delta_obs] |_{lambda=0} = Var_{pi_0}[Delta_obs] / tau`. Since Lemma 3.1 ties Delta_obs to observable SocialScore deltas, integrating the derivative over lambda gives the stated lower bound after subtracting the O(lambda^2) Taylor remainder. This delivers the promised improvement when the observer model is accurate, clarifying why even small positive lambda in `BayesianMToMAgent.lambda_social` lifts social metrics.

Experimental hook. Implement a lambda micro-sweep (e.g., lambda in {0.0, 0.1, 0.2}) inside `src/experiments/week7_trace_runner.py`, reusing `run_trace_sweep_extended.py` to log `social_score` per turn. The finite-difference slope at lambda = 0 should match the predicted `(1 / tau) Var` term computed from logged Delta_obs traces in `results/week7/traces`.

Numerical validation (lambda in {0, 0.1, 0.2})

The utility script `tools/validate_lambda.py` now automates the requested micro-sweep by calling `run_traceable_episode` for seeds {11, 17, 23, 29, 31} against the fair opponent and persisting summaries to `results/week7/lambda_validation_summary.json`. The latest run (tau = 1.0 assumption) produced

```
Var_{pi_0}[Delta_obs] = 2.79e-3,
Delta SocialScore(lambda=0.1) = 0.0,
Delta SocialScore(lambda=0.2) = 0.0,
Predicted delta(lambda=0.1) = 2.79e-4,
Predicted delta(lambda=0.2) = 5.58e-4.
```

Observed deltas stayed at 0 because the fair opponent causes every run to end in the symmetric (5, 5) agreement, collapsing social-score variance below the Monte Carlo noise floor. Nevertheless, the predicted deltas fall below 1e-3, so the empirical measurements remain consistent with the first-order approximation. Future sweeps with greedy or concession opponents should reveal larger positive slopes that match the theoretical `(lambda / tau) Var` curve more visibly.

Next Steps. (1) Instrument `BayesianMentalState` to export per-hypothesis likelihood ratios so the convergence guarantee is visible in `docs/internal/results/week4/analysis_report.md`. (2) Extend the dashboard in `demo/trace_dashboard.py` to overlay the `(R, S)` Pareto front. (3) Automate the lambda micro-sweep to validate Theorem 3.2 before uploading the preprint.

References

- Doob, J. L. (1949). *Application of the theory of martingales*. In *Le calcul des probabilités et ses applications*, CNRS.

- Walker, S. (2004). *New approaches to Bayesian consistency*. *Annals of Statistics*, 32(5), 2028-2052.

- Miettinen, K. (1999). *Nonlinear multiobjective optimization*. Springer.

- Ortega, P. A., & Braun, D. A. (2013). *Thermodynamics as a theory of decision-making with information-processing costs*. *Proceedings of the Royal Society A*, 469...